UCRL-CR-136348
B324163

Scaling and Performance of a 3-D Radiation Hydrodynamics Code on Message-Passing Parallel Computers: Final Report

J.C. Hayes and M.L. Norman

October 28, 1999

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

Scaling and Performance of a 3-D Radiation Hydrodynamics Code on Message-Passing Parallel Computers: Final Report

John C. Hayes
Laboratory for Computational Astrophysics &
National Center for Supercomputing Applications
University of Illinois, Urbana-Champaign

Michael L. Norman
Laboratory for Computational Astrophysics, Department of Astronomy &
National Center for Supercomputing Applications
University of Illinois, Urbana-Champaign

1. Introduction

This report details an investigation into the efficacy of two approaches to solving the radiation diffusion equation within a radiation hydrodynamic simulation. Because leading-edge scientific computing platforms have evolved from large single-node vector processors to parallel aggregates containing tens to thousands of individual CPU's, the ability of an algorithm to maintain high compute efficiency when distributed over a large array of nodes is critically important. The viability of an algorithm thus hinges upon the tripartite question of numerical accuracy, total time to solution, and parallel efficiency.

In this study we evaluate and compare the performance of implicit and explicit solutions to radiation diffusion within three-dimensional simulations of diffusion through static media, and of diffusion coupled to hydrodynamic motions. The implicit method may employ a diagonally-preconditioned Conjugate Gradient (CG) or a multigrid (MG) linear system solver. As of this writing, the MG package is not yet robust enough to handle the special challenges resulting from the strongly discontinuous diffusion coefficients in our test problem, but it has been used on a variety of other problems and continues to be improved, with the aim of eventually applying it to the ICF test described in this report. The explicit method uses the unconditionally stable Product Formula (PF) formalism to evaluate the diffusion term in the radiation energy equation. Unlike traditional explicit methods, the PF method does not place stability restrictions upon timestep growth, although we shall see that restrictions motivated by accuracy constraints do exist. Additionally, the explicit nature of the calculation obviates the need for inverting a matrix (or performing the equivalent), as required by implicit methods. Finally, the locality of the explicit solution sharply reduces the required inter-node communication during a solution step.

In contrast, the implicit solution demands both a matrix solution and intensive inter-node communication owing both to the need for ghost-zone data and for global reduction operations used in convergence checks. Opposite these concerns is the fact that the implicit method allows for timestep growth which is not Courant limited, and indeed far more robust than that allowed by the PF in the applications detailed here.

The ultimate objective of this report is to ascertain the relative merits of the implicit and unconditionally stable explicit approaches to radiation diffusion. We shall see that the PF algorithm possesses outstanding scalability over numbers of nodes of up to at least order 10^3, but that the current operator splitting scheme used to implement the PF introduces an accuracy constraint which is far too severe for the challenging test problem posed in this study. The prospects for alleviating this constraint with an alternate scheme and the benefits (if any) of doing so are discussed in the summary section of this report.

This report is organized in the following manner: §2 describes the physical problem used to benchmark the code and lists the various numerical tests derived from the basic problem. §3 provides an overview of ZEUS-MP and lists the equations used to model radiating flows. §4 describes the mathematical solution techniques employed in the implicit and explicit radiation solvers, with full details of the PF deferred to Appendix B. §5 explores the dependence of the accuracy of each algorithm upon timestep control and operator splitting. §6 presents the suite of scaling results, which come in the form of "scaled work" tests, where the number of grid points scales (linearly) with the number of nodes, and "fixed work" tests, where a problem of fixed total size is distributed across a varying number of nodes. §7 presents a grand summary and discussion of all results presented herein.

2. Test Problems

2.1. The Physical Model

To provide meaningful answers to the questions posed above, we challenge the various algorithms in ZEUS-MP through the construction of a three-dimensional, ablation-driven implosion. The physical characteristics of the problem are inspired by Inertial Confinement Fusion (ICF) tests, though we must note at the outset that some elements of physics central to real ICF simulations are excluded here (e.g., nuclear burning, non-ideal and/or multi-species fluids, multi-frequency radiation transport). We consider a spherical "capsule" with the following characteristics:

- DT gas within $0 \leq R \leq 0.087$ cm, $\rho = 0.025$ g/cc;
- DT ice from $0.087 < R \leq 0.095$ cm, $\rho = 0.25$ g/cc;
- CH foam from $0.095 < R \leq 0.111$ cm, $\rho = 1.2$ g/cc;
- He gas from $0.111 < R \leq 0.497$ cm, $\rho = 0.01$ g/cc; and
- Au for $R > 0.497$ cm, $\rho = 19.3$ g/cc.

We model this capsule on a grid constructed as follows:

- $0.01 \leq R \leq 0.490$ cm; reflecting inner and outer boundary conditions.
- $\pi/4 \leq \theta \leq 3\pi/4$; periodic boundary conditions.

- $\pi/4 \leq \phi \leq 3\pi/4$; periodic boundary conditions.
- Grid resolutions ranging from 256x16x16 to 512x256x256

The following physical inputs are imposed:

- Single-fluid ideal gas EOS with $\Gamma = 5/3$.
- Opacities derived from an analytic power-law formula for the mean-free-path: $\lambda = \lambda_o \times (\rho/\rho_o)^{-\nu} \times (T/T_o)^n$; $(\nu, n > 0)$.

Additionally, we assume grey radiation transport. The assumption of a single-component fluid deserves special attention, as it has mandated several compromises in the construction of the test problem. From a hydrodynamic perspective, the real materials used in an ICF capsule obey different equations of state with different effective "Γ's." While the gaseous components of the capsule might reasonably be approximated as ideal gases, other components (most notably the DT ice and the gold) possess material strength which can not be modeled in our prescription. For this reason, the outer radial boundary of our grid is a reflecting one that lies just inside the inner radius of the gold layer. This eliminates the spurious effect of ablation at a helium-gold interface, a difficulty encountered during early trials of the test problem.

The consequences of the single-fluid approximation for the construction of opacities (both Rosseland and Planck means) are dramatic and far-reaching. The opacities used in ICF simulations can be approximated by power-law expressions of a Kramer's Law type, where

$$\kappa = \kappa_o \left(\frac{\rho}{\rho_o}\right)^2 \left(\frac{T}{T_o}\right)^{-3.5} . \qquad (1)$$

The crucial distinction between materials lies in the values of the normalization constants chosen, which vary drastically from one material to the next. In the initial model for our simulations, the source (He) temperature is 12,000 times larger than that of the target region (300 vs 0.025 eV). Furthermore, early versions of the problem possessed densities at the He-CH interface which changed by a factor of 2200 (0.0005 g/cc for the He, vs. 1.1 g/cc for the CH foam). Naive use of equation (1) with such large factors leads to contrasts (21 orders of magnitude!) in the material opacity which are far higher than those encountered (roughly 5 orders of magnitude) in real ICF materials. Modeling such (ρ,T) conditions with a single-fluid opacity law of the Kramer's form has two profound consequences: (1) For any remotely physical choice of opacity in the source region, the CH foam layer becomes so optically thick that ablation on physically relevant timescales is completely suppressed, thus destroying the desired physical character of the problem. (2) The extreme variation in opacity leads to an extreme variation in the value of diffusion coefficients. This leads to matrices – for an implicit solution – that become profoundly ill-conditioned once timesteps become dynamically meaningful. Such occurences repeatedly doomed the implicit linear system solver.

To successfully implement the test problem, we derived opacities from the following expression for the mean-free path:

$$\lambda(\mathrm{cm}) \;=\; 10^{-6}\left(\frac{\rho}{1.2\mathrm{gcm}^{-3}}\right)^{-2}\left(\frac{T}{0.025eV}\right)^{1.2}, \tag{2}$$

with the restriction that $10^{-6} \le \lambda(cm) \le 0.1$. With this relation for the mean-free path, we then define analytic Rosseland and Planck mean opacities (in units of cm^{-1}) as follows:

$$\kappa_\mathrm{R} \;=\; \frac{1.0}{\lambda}; \tag{3}$$

$$\kappa_\mathrm{P} \;=\; 10.0\kappa_\mathrm{R}. \tag{4}$$

These choices allowed us to simultaneously meet the following goals: (1) retaining the ablation effect, (2) avoiding the creation of pathologically ill-conditioned matrices, (3) allowing radiation diffusion to proceed on dynamically relevant timescales, while at the same time (4) reducing the amount of radiative "preheating" of the DT core as much as possible.

2.2. The Test Suite

With a physical problem constructed as outlined above, we may impose a battery of tests that gauge the performance and accuracy both of the integrated code and of selected modules. To begin, we examine tests involving nothing more than pure diffusion of a radiation front through a medium that is both static and thermally frozen; i.e. both hydrodynamic and matter-radiation coupling terms are zeroed. This allows a pure head-to-head comparison of the implicit and explicit diffusion solvers with regard to all aspects of performance. Next, we retain the assumption of a static medium, but we allow thermal communication between the radiation and material. In this set of tests the limitations of the operator-splitting scheme used to implement the PF will become clearly visible. Indeed we will see that accurate solutions involving matter-radiation coupling are unattainable with the PF for any reasonable choice of timestep restriction parameters, and we will show that this is a consequence of the operator splitting scheme itself rather than an inherent flaw with the PF solution. Nonetheless, we will proceed to full radiation hydrodynamic solutions where the explicit and implicit algorithms are compared with regard to approximate time for solution, with an aim toward ascertaining whether an *accurate* implementation of the PF algorithm could, even in principle, be superior to an implicit scheme using an efficient solver. Results of these comparisons will strongly suggest that continued research effort may be more profitably directed toward performance-enhanced implicit schemes than explicit methods.

3. ZEUS-MP and the Equations of Radiation Hydrodynamics

Both radiation solvers are deployed within ZEUS-MP, the message-passing version of the ZEUS algorithm (Stone & Norman, 1992a,b; Stone, Mihalas, & Norman, 1992). The equations solved in ZEUS-MP may be written as follows:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot v = 0 \tag{5}$$

$$\rho\frac{Dv}{Dt} = -\nabla p - \nabla\cdot Q + \frac{1}{c}\chi F \tag{6}$$

$$\frac{De}{Dt} = -p\nabla\cdot v - 4\pi\kappa_{\rm P}B + c\kappa_{\rm P}E - Q : \nabla v \tag{7}$$

$$\frac{DE}{Dt} = -\nabla\cdot F - \nabla v : P + 4\pi\kappa_{\rm P}B - c\kappa_{\rm P}E \tag{8}$$

$$F = -\frac{c\Lambda(E)}{\chi}\nabla E \tag{9}$$

In the above, χ is the sum of the Rosseland mean opacity and the scattering opacity, which is taken to be zero in all tests discussed here. In the flux equation, $\Lambda(E)$ is the flux-limiter. We have chosen a Levermore-Pomraning flux-limiter for all tests.

The ZEUS codes employ a Wilson-LeBlanc scheme for solving the hydrodynamic equations on an Eulerian grid; ZEUS-MP uses the Message Passing Interface (MPI) standard for distributing the calculation across multiple processors. The original version of ZEUS-MP, developed by Robert Fiedler, contained modules for gas hydrodynamics on arbitrary orthogonal grids and explicit radiation diffusion on cartesian grids only. A detailed report on the performance of this code on a wide variety of platforms is available (Fiedler 1997).

The current version of ZEUS-MP adds a generalized version of the PF algorithm for arbitrary orthogonal grids, and an implicit radiation diffusion solver which can use either a preconditioned CG or an MG linear system package. The entire code has been designed in a highly modular way which allows for alternate routines (or even operator splitting schemes) to be swapped with relative ease.

In the operator-splitting schemes currently employed, the PF is used to evaluate the flux divergence term alone. The remaining terms in the coupled gas and radiation energy equations (eqs. 7 and 8) are solved in the remaining two stages of a 3-stage splitting scheme. In the implicit method, all terms in the radiation energy equation are solved with the iterative linear system solver, and the material temperature is then updated in the second stage of a two-stage splitting scheme. A cursory description of the PF formalism and details of the operator splitting schemes used in the explicit and implicit algorithms are presented in the following section. A full mathematical treatment of the PF approach is detailed in Appendix B, and the precise functional values of the matrix elements in the implicit solution are documented in Appendix C.

4. The Diffusion Solvers

4.1. Implicit Diffusion

4.1.1. Mathematical Formalism

We begin the discussion of the implicit solution by considering the operator splitting scheme used to update the radiation and gas energy densities. In ZEUS-MP, artificial viscosity and PdV terms are handled in separate sections of the code. Thus we consider reduced forms of equations (7) and (8) written in time-centered form as follows:

$$E^{n+1} - E^{n} = \Delta t \left[\kappa_P \left(4\pi B - cE^{n+1}\right) - \nabla \cdot F - \nabla v : P\right] \qquad (10)$$

$$e^{n+1} - e^{n} = \Delta t \left[-\kappa_P \left(4\pi B - cE^{n+1}\right)\right] \qquad (11)$$

In what follows, we will make use of the following relations and definitions:

- $B = caT^4/\pi$.
- $e = \rho c_v T$.
- $k_r \equiv c\kappa_P \Delta t$.
- $\mathcal{F} \equiv \Delta t \nabla \cdot v : f$, where f is the Eddington tensor. ($P = fE$.)
- $[E]_D \equiv \Delta t \nabla \cdot (D \nabla E)$, where D is the flux-limited diffusion coefficient.

We will also simplify the time-centering notation by replacing "$n+1$" with a "prime" symbol and by dropping the "n" superscript altogether.

The splitting scheme employed parallels that used in the 3-T ARES code at LLNL (Baldwin *et. al* 1998). To proceed, we will make use of the temperature dependence of the Planck function and gas energy to write an equation for the gas temperature. The essential feature of this approach is that we linearize the source function with the following relation:

$$T'^4 \simeq T^3 (4T' - 3T). \tag{12}$$

We may therefore use (12) to transform (11) as

$$T' = \left(\frac{1}{\Delta}\right) \left[T + \left(\frac{k_r}{\rho c_v}\right) \left(3aT^4 + E'\right)\right], \tag{13}$$

where we have defined Δ as

$$\Delta \equiv 1 + \left(\frac{4ak_r}{\rho c_v}\right) T^3. \tag{14}$$

Use of (13) for T', along with the T-dependent expressions for e and B, allows us to write (10) as

$$\left[1 + \frac{k_r}{\Delta} + \mathcal{F}\right] E' - [E']_D = E + \frac{k_r a T^4}{\Delta}. \tag{15}$$

(15) has the form of a matrix equation, $\mathbf{A}\vec{x} = \vec{b}$, if we identify $\vec{b}$ as the RHS of (15), $\vec{x}$ as the solution vector, E', and $\mathbf{A}$ as

$$\mathbf{A} \equiv \mathbf{1} + \frac{k_r}{\Delta} + \mathcal{F} - [E]_D, \tag{16}$$

where in (16) it is understood that $[E]_D$ represents the diffusion operator that operates on E'. All off-diagonal elements of $\mathbf{A}$ are contributed by the diffusion operator; the other terms contribute strictly to the diagonal.

Armed with the above, an implicit update to the gas and radiation energy densities proceeds as follows: (1) using opacities and diffusion coefficients calculated with the old matter temperature, update E with a matrix solution to (15). (2) With the updated E, use the analytic expression (13) to update T, and the constitutive relation for $e(\rho, T)$ to update e.

4.1.2. The Implicit Linear System Solvers

The bulk of the implicit tests in this report are performed with the CG linear system solver. The theory of the Conjugate Gradient method will not be reproduced here, as there are many sources available on the subject (e.g. Barrett *et. al*, 1994). However, a few features specific to this algorithm are worth mentioning. Currently, the only preconditioner available in our algorithm is diagonal preconditiong, but the package has been written, as has the parent code, in a highly modular way which allows other preconditioning options to be implemented with little or no disturbance to the driver routine. The CG solver has also been designed from the outset to exploit machine architectures which support simultaneous communication and computation. To this end, all MPI communication requests are non-blocking, and the Fortran DO loops have been structured so that work on data points away from local grid boundaries may proceed while ghost zone data is being exchanged. The scaling tests presented in §6 will quantify the degree to which we have succeeded. A third feature deserving special notice is the fact that the CG package has been designed to accommodate matrices generated by 'multigroup' (multiple energy group) equations. This feature is not yet present in the MG linear system package, described below.

The MG solver, called MGMPI, was originally developed by James Bordner as an autonomous package for solving the Poisson equation. Recently, it has been modified for installation within the ZEUS-MP implicit diffusion driver as an alternate package for solving the radiation diffusion package. The current MGMPI library contains 3 independent driver routines: one specifically designed for the Poisson equation, and two which handle general elliptic PDE's that have been discretized on arbitrary orthogonal meshes. Of the latter two, one driver is designed for symmetric matrices and the other for non-symmetric matrices. Both of these routines are written to perform solutions on matrices which are computed externally and input through the subroutine calling arguments, in contrast to the Poisson routine which generates its matrix internally.

MGMPI has been written to accept orthogonal coordinate meshes which use the covariant metric coefficients defined by ZEUS-MP and used throughout the hydro routines. The solver accepts Neumann, Dirichlet, and periodic boundary conditions. In contrast to the CG solver, MGMPI's MPI communication calls are synchronous (blocking); thus computation and communication may not proceed simultaneously.

A gzipped tar file containing the MGMPI source code and Makefile scripts may be downloaded from the World Wide Web at the following address: http://www.ncsa.uiuc.edu/ ~bordner/mgmpi.html. This website also contains a user guide and separate documents describing algorithms and performance. All three documents are updated periodically as modifications to the solver are made.

4.2. Explicit Diffusion: The Product Formula

The second basic approach to implementing radiation diffusion in ZEUS-MP is based on an explicit, yet unconditionally stable, update to the diffusion term in equation 8. The method of solution is based upon the Product Formula (PF); an implementation of the PF for computing diffusion on uniform cartesian grids is described in Graziani (1995). The routine in ZEUS-MP is a generalized version of the approach discussed in the Graziani paper. Full documentation of our formalism appears in Appendix B, but a brief summary of the method is appropriate here. Consider a multi-stage operator splitting scheme where the first update to the radiation energy density accounts for the diffusion term alone:

$$\frac{dE}{dt} = \nabla \cdot (D\nabla E). \tag{17}$$

We may write this symbolically as a matrix equation:

$$\frac{dE_i}{dt} = \Lambda_{ij} E_j, \tag{18}$$

where Λ is a matrix operator containing all information about diffusion coefficients, grid coefficients, and boundary sources. In the limit of small timesteps, the solution to (18) is

$$E_i(t + \Delta t) = \exp(\Lambda_{ij}\Delta t) E_j(t). \tag{19}$$

The art of implementing the PF solution lies in the manner in which the Λ matrix is exponentiated. In practice, Λ is decomposed into a sum of two (almost) block-diagonal matrices which can be exponentiated with relatively minor effort. In general, there is no unique decomposition for the Λ matrix, and issues remain regarding the advantages (if any) of one choice over another. Our choice, along with values of the matrix elements which obtain for various types, is documented extensively in Appendix B.

Since the PF is used to evaluate the contribution from the flux gradient term alone, it follows at once that the operator splitting scheme detailed in §4.1.1 may not be applied here. Rather, the update of E is split into a pair of substeps before the final matter temperature update is performed. Recalling that we may write the radiation energy equation, in "clean" notation, as

$$E' - E = k_r \left(aT'^4 - E'\right) + [E']_D - \mathcal{F}E', \tag{20}$$

we delineate a two-step solution as

$$E^* - E = [E]_D, \tag{21}$$

with the PF providing an explicit update from the diffusion term, and

$$E' - E^* = k_r \left(aT'^4 - E'\right) - \mathcal{F}E'. \tag{22}$$

The solution to (21) is straightforward, and – as shall be seen in §6 – far less expensive with regard to CPU cost per timestep, as compared with the implicit method. After linearizing the source term in (22) via equation 12, (22) may be transformed as

$$E' = \frac{E^* + \frac{k_r a T^4}{\Delta}}{1 + \frac{k_r}{\Delta} + \mathcal{F}}. \tag{23}$$

Therefore, to update both the gas and radiation energy densities, we employ a 3-stage scheme where the radiation energy density is advanced in time via (21) and (22), and the gas energy density is updated, as in the implicit scheme, with (13) and the equation of state.

5. Timestep Control, Operator Splitting, and Solution Accuracy

Our ultimate goal of this project is to compare the costs of different algorithms for computing solutions at a given accuracy. While scalability (however it is defined) plays a pivotal role in such determinations, the number of timesteps required for a completed solution is equally relevant. The number of timesteps required is a direct function of the degree to which physical variables may change in one timestep, while still evolving toward an acceptable solution. In this study, we define "acceptable" as producing quantities that differ from asymptotically converged counterparts at approximately the ten percent level. In our studies we have used both spatial profiles and time-evolution profiles of the relevant variables (density; gas and radiation temperatures) as accuracy gauges.

5.1. Tests in Static Media

For test cases which exclude material motion, the runs were performed in 1-D for the sake of economy. Because the physical configuration is spherically symmetric by decree, we may use 1-D results in these cases without loss of generality. All hydrodynamic tests were carried out in 3-D. Because the 3-D runs involved grids with 256 radial zones, the 1-D results presented here were also run with 256 zones. We found that the timestep controls producing "converged" solutions were the same at both 256 and 480 zones. The accuracy of the higher resolution runs degrades somewhat more rapidly with relaxed timestep control than do the lower resolution runs, but the difference is not dramatic. Because the 3-D runs used 256 radial zones, we will use accuracy criteria based on that radial resolution as our guide.

While the tests of ultimate interest are those involving full radiation hydrodynamics, we also examine scaling and timing tests involving diffusion only. Therefore we document the allowed tolerances, for both the implicit and explicit algorithms, producing solutions which are accurate at the 10% level. In the case of the explicit PF algorithm, the timesteps for diffusion alone are controlled by specifying a maximum allowed ratio of the timestep to the radiation Courant time, defined as $C_{rad} = \frac{1}{4}\frac{\Delta x^2}{D}$, where D is the local diffusion coefficient, and Δx is a local zone width. We find that a "converged" solution is reached at a radiation Courant number of 0.5, and an "acceptable" solution can be retained at a C_{rad} of 10. Figure 1 shows profiles for the two cases at an evolution time of 10^{-9} seconds. The solid line and dashed lines give solutions obtained with C_{rad} equal to 0.5 and 10.0, respectively. The maximum deviation from the converged solution occurs at the center of the capsule (at the origin of the plot), and has a relative error of 9.6%.

Figure 2 shows the corresponding plot for the case of evolution computed with the implicit CG solver. Here, timesteps are regulated by restricting the maximum fractional change in the radiation energy density (per timestep) to lie within a specified tolerance. In figure 2, the converged solution (solid line) is obtained with a maximum tolerance of 0.005, and the dashed line shows the solution obtained with a tolerance of 0.1. Remarkably, the implicit solution shows little additional deterioration of the solution for tolerances beyond 0.1; indeed the maximum relative error had grown to only 4% for a tolerance of 0.3, and at 0.4 the solver encountered convergence difficulties. This unusual behavior is not seen in tests which include matter-radiation coupling and/or hydrodynamics, and in more sophisticated tests a tolerance of no more than 10% was allowed.

The pure diffusion tests compute the evolution of the radiation field according to the diffusion term alone. In the following tests we compute radiation diffusion with absorptivity/emissivity terms included, and we allow evolution of the material temperature to proceed under the action of these same terms. We retain the assumption of a static medium, so the radiation stress term in (8) and the pdV and viscosity terms in (7) remain inactive. In the explicit runs, timesteps are controlled by using C_{rad} to control the evolution of E and a restriction on $\Delta e/e$ to control the evolution of the e. In the implicit runs, restrictions on the fractional change in both E and e are used. The same tolerance value is used for both variables in the following results.

With the implicit algorithm, an asymptotically correct solution is reached for a maximum tolerance of 0.005, and solutions accurate at the 10% level are achievable with a tolerance as high as 0.05, a factor of 2 lower than in the pure diffusion case. This behavior is shown in figures 3 and 4, where we compare the radiation and gas temperature profiles, respectively, for both tolerances. The sharp dip in the gas temperature between 0.9 and 1.0cm is due to the presence of the dense carbon foam layer, with its higher density and correspondingly higher heat capacity.

While accurate solutions for the gas and radiation temperatures are achievable with the

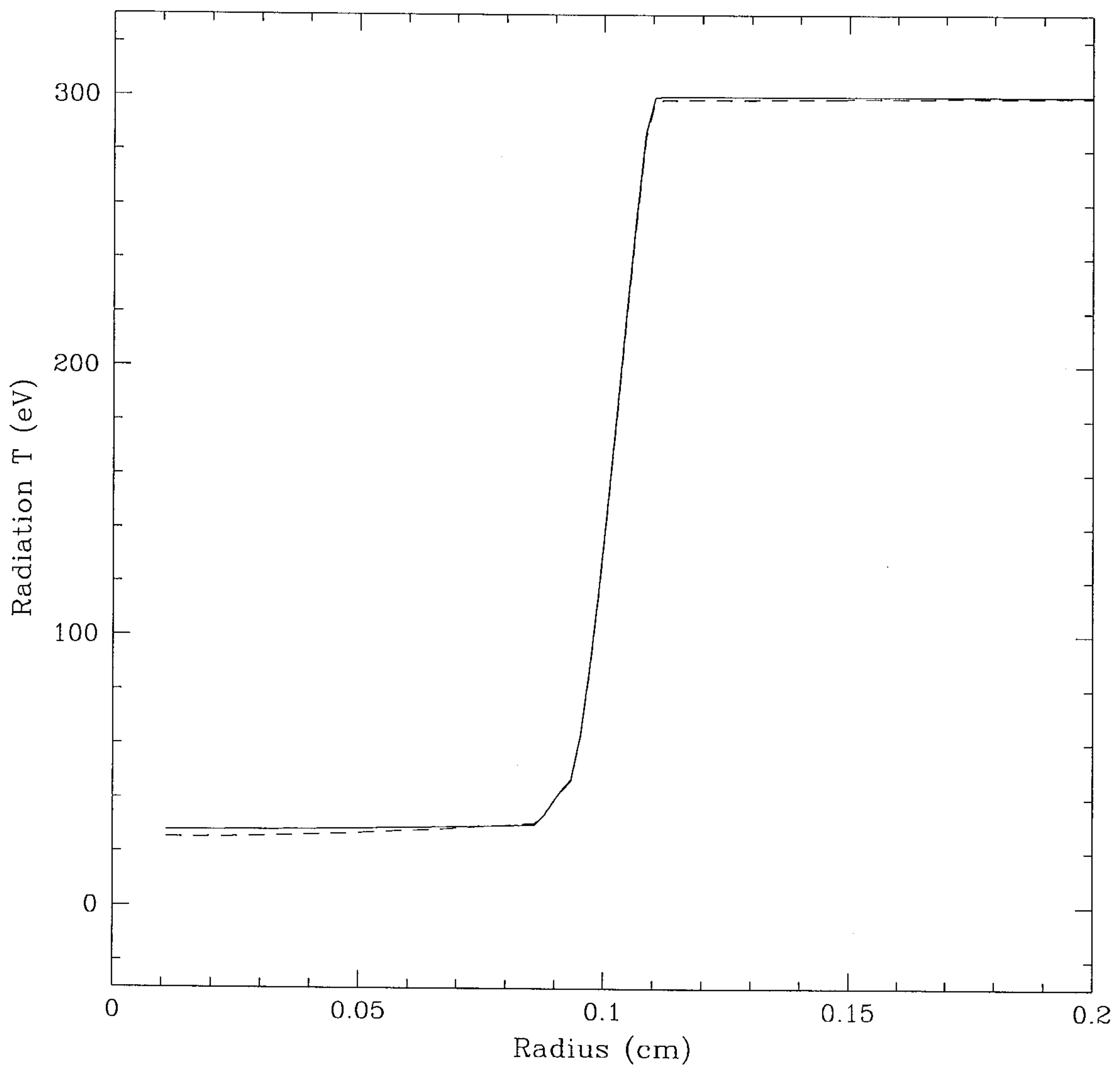

Fig. 1.— Comparison of pure diffusion runs with the PF for $C_{rad} = 0.5$ (solid line) and 10.0 (dashed line).

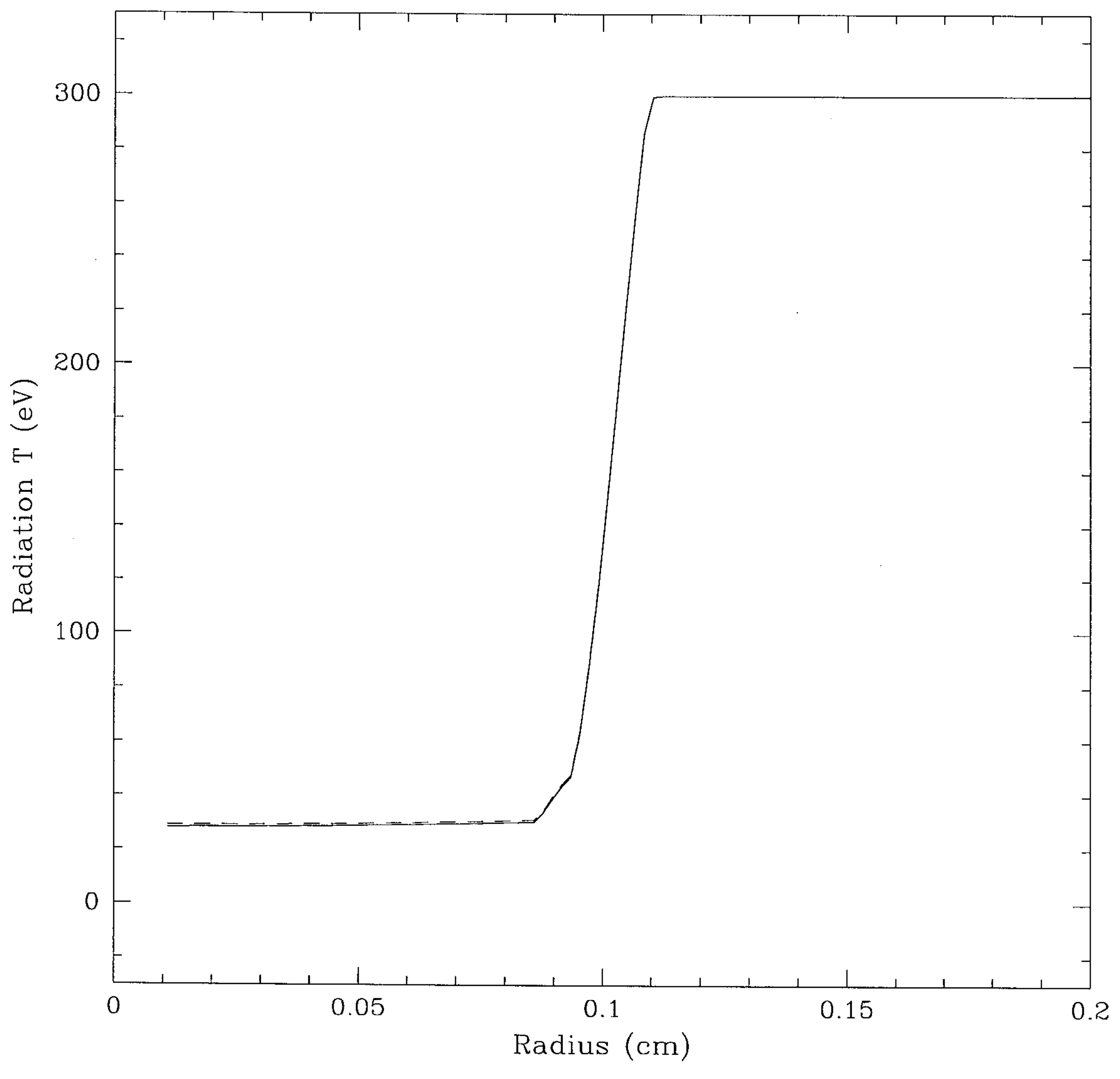

Fig. 2.— Comparison of pure diffusion runs with the implicit CG solver for $\Delta E/E = 0.005$ (solid line) and 0.1 (dashed line).

implicit solver using reasonable values of the restriction parameters, such is manifestly not the case with the explicit algorithm. Examination of figures 5 and 6 reveals that accuracy is approached only with $C_{rad} \leq 0.005$, and the solution is hopelessly wrong for a C_{rad} of only 0.5! In both cases, the tolerance on the relative change in e was 0.05, but for both values of C_{rad}, the asymptotic timestep is determined entirely by the Courant factor.

The behavior of both the gas and radiation temperatures is consistent for the two algorithms in the converged cases, but the explicit results are disastrously incongruous for even a semi-reasonable choice of C_{rad}. Such behavior was not anticipated, and we are compelled to find the cause. The fact that both the gas and radiation temperatures approach the correct values in the limit of extremely small timesteps ($\sim 10^{-16}$ sec!) indicates that an outright error in either the PF, radiation source/sink, or matter temperature algorithms is not to blame. We have already seen that the PF produces correct results when used by itself, and tests involving pure heating and cooling without diffusion have yielded results precisely consistent with implicit tests. The matter temperature algorithm, as discussed in §3, is the same in the implicit and explicit routines. We are therefore led to suspect that the operator splitting, which differs in the two approaches, is the culprit. Recall that in the explicit scheme, we evaluate diffusion and absorption/emmission in two stages rather than one because the PF handles diffusion exclusively. If the relative values of $\dot{E}$ from these two contributions are sharply different, then operator splitting in this manner could be problematic.

Both analytic and empirical estimates of these relative contributions show that the difference between the two is indeed extreme. Consider a cold zone of width $\sim 10^{-3}$ cm, bounded by an identical cold zone on the left and a hot zone ($E_{hot} >> E_{cold}$) on the right. The value of the diffusion term in the middle zone may be estimated, to an order of magnitude, as

$$\dot{E} \sim \frac{E_{hot}}{\Delta r^2} \frac{c}{3\chi}, \tag{24}$$

which gives a value of roughly 10^{22} erg s^{-1} for appropriate values of the physical parameters at the interface between the helium source and carbon foam layer. In contrast, the middle cold zone is initially in local thermal equilibrium, so the starting value of $\dot{E}$ is effectively zero. Recalling that the temporal change in E is controlled by the radiation Courant time ($\sim 10^{-13}$ sec), we see that the change in E due to diffusion could easily overwhelm that due to coupling in our operator splitting scheme. Monitoring the change in E, using ZEUS-MP, from the diffusion term affirms that this is the case. At a stage when the fractional change in the matter temperature is on the order of 10^{-3}, the fractional change in E from the diffusion term, as controlled by C_{rad} is six orders of magnitude higher. The subsequent adjustment of E due to the source/sink terms is comparable, but not exactly equal to, that of the diffusion term. Even a small systematic inaccuracy could lead to an enormous error over significant

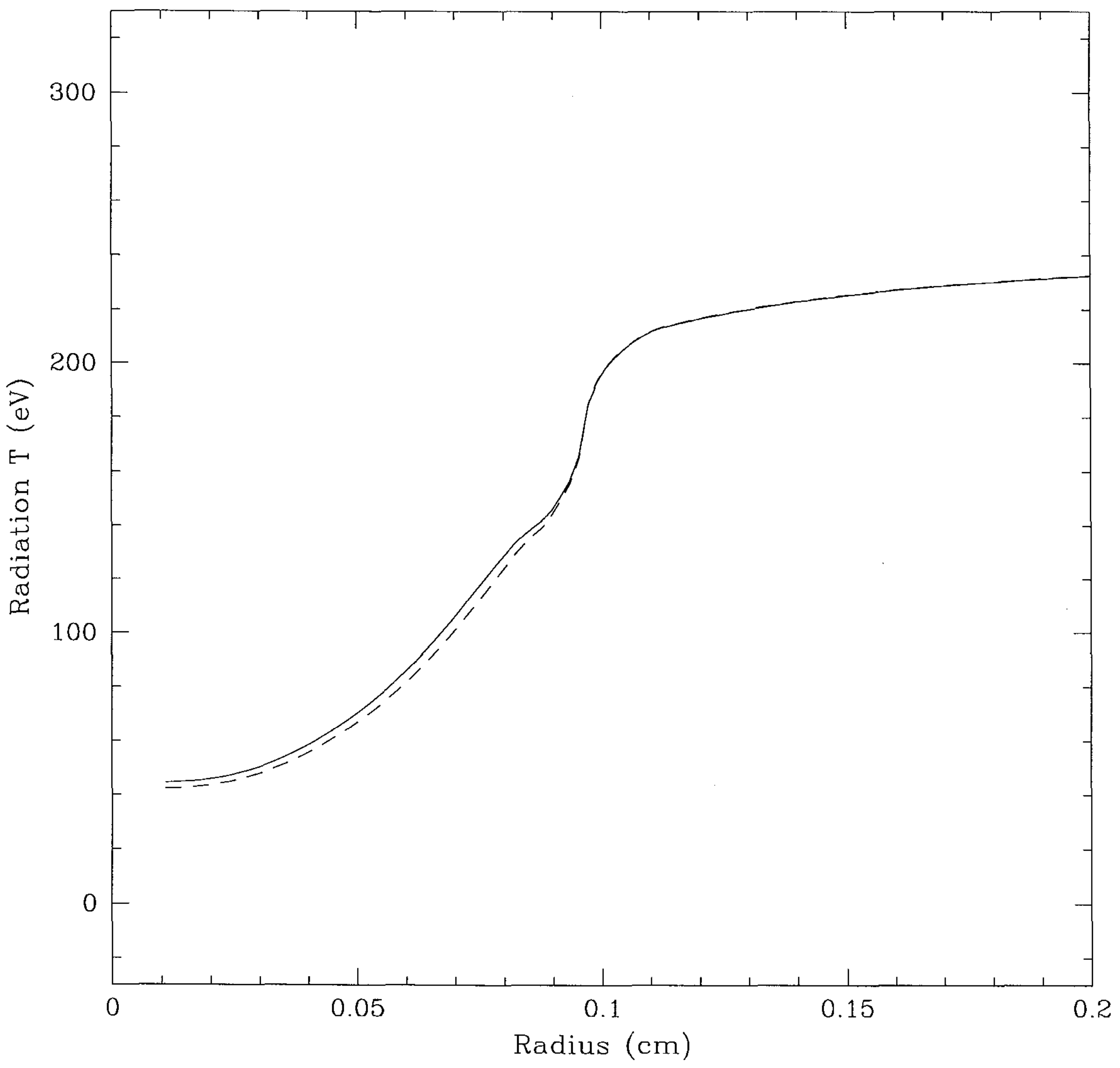

Fig. 3.— Diffusion with coupling: comparison of the radiation temperature from the implicit CG solver with $\Delta E/E = 0.005$ (solid line) and 0.05 (dashed line).

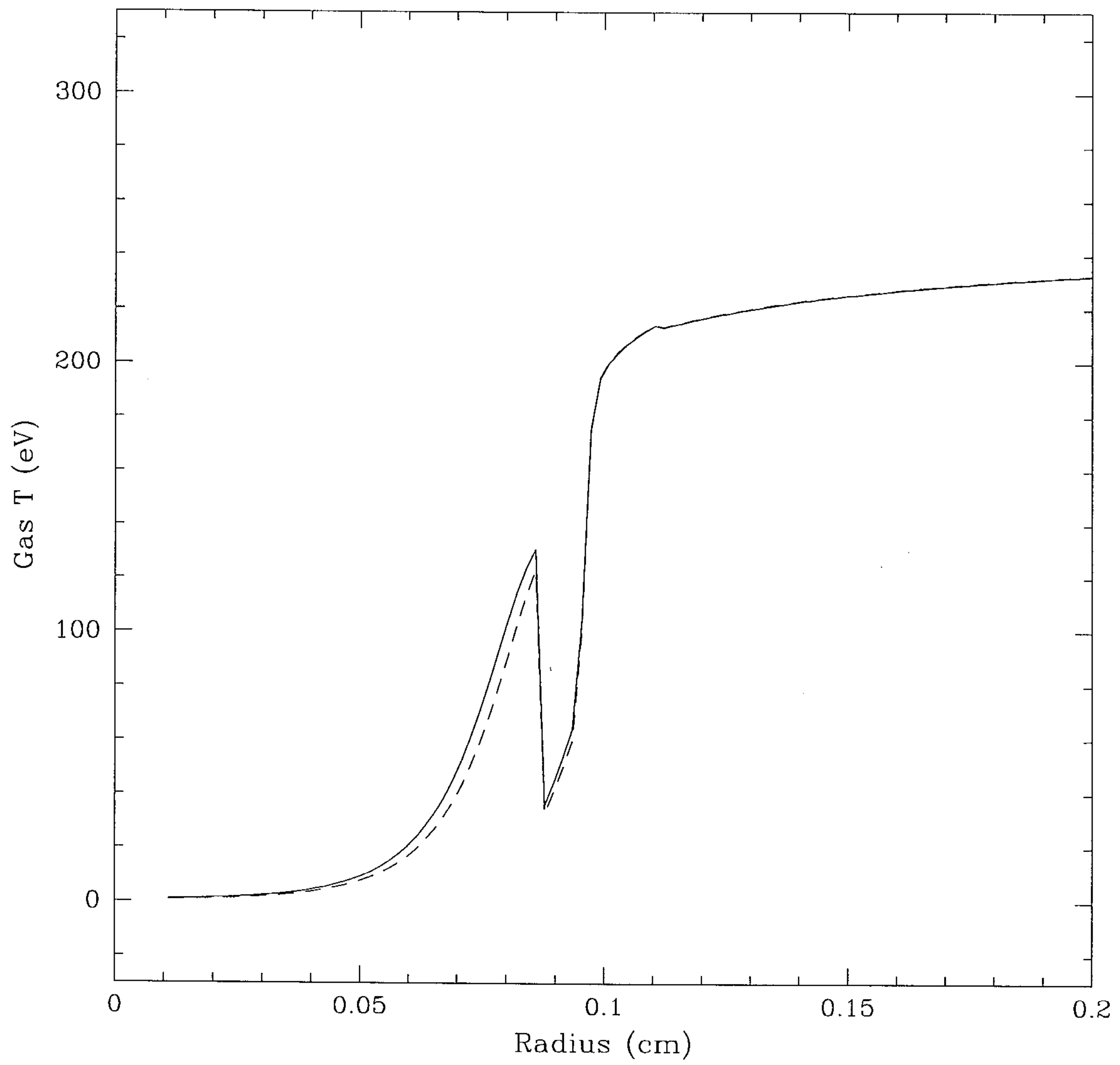

Fig. 4.— Diffusion with coupling: comparison of the gas temperature from the implicit CG solver with $\Delta E/E = 0.005$ (solid line) and 0.05 (dashed line).

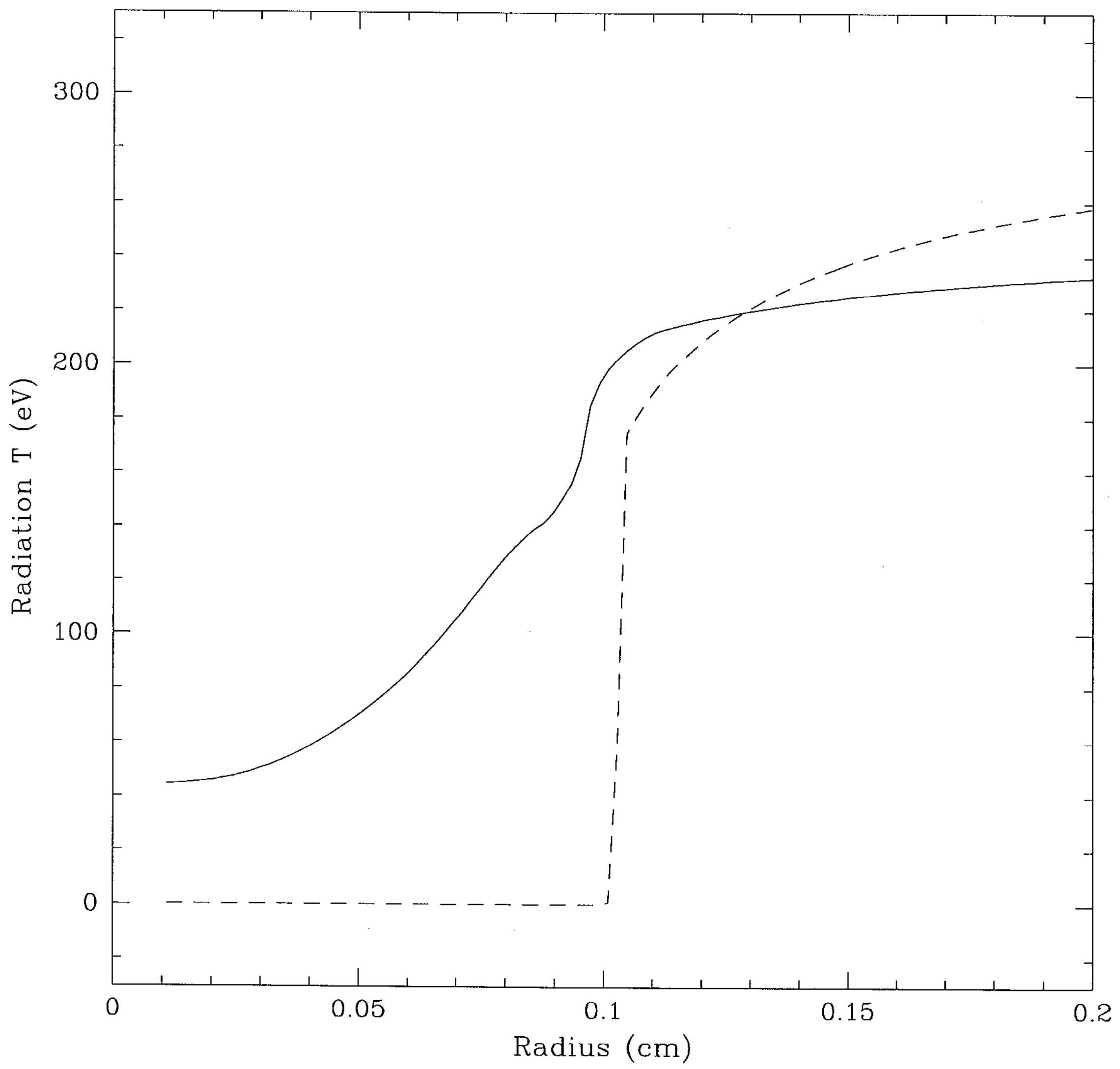

Fig. 5.— Diffusion with coupling: comparison of the radiation temperature from the PF with $C_{rad} = 0.005$ (solid line) and 0.5 (dashed line).

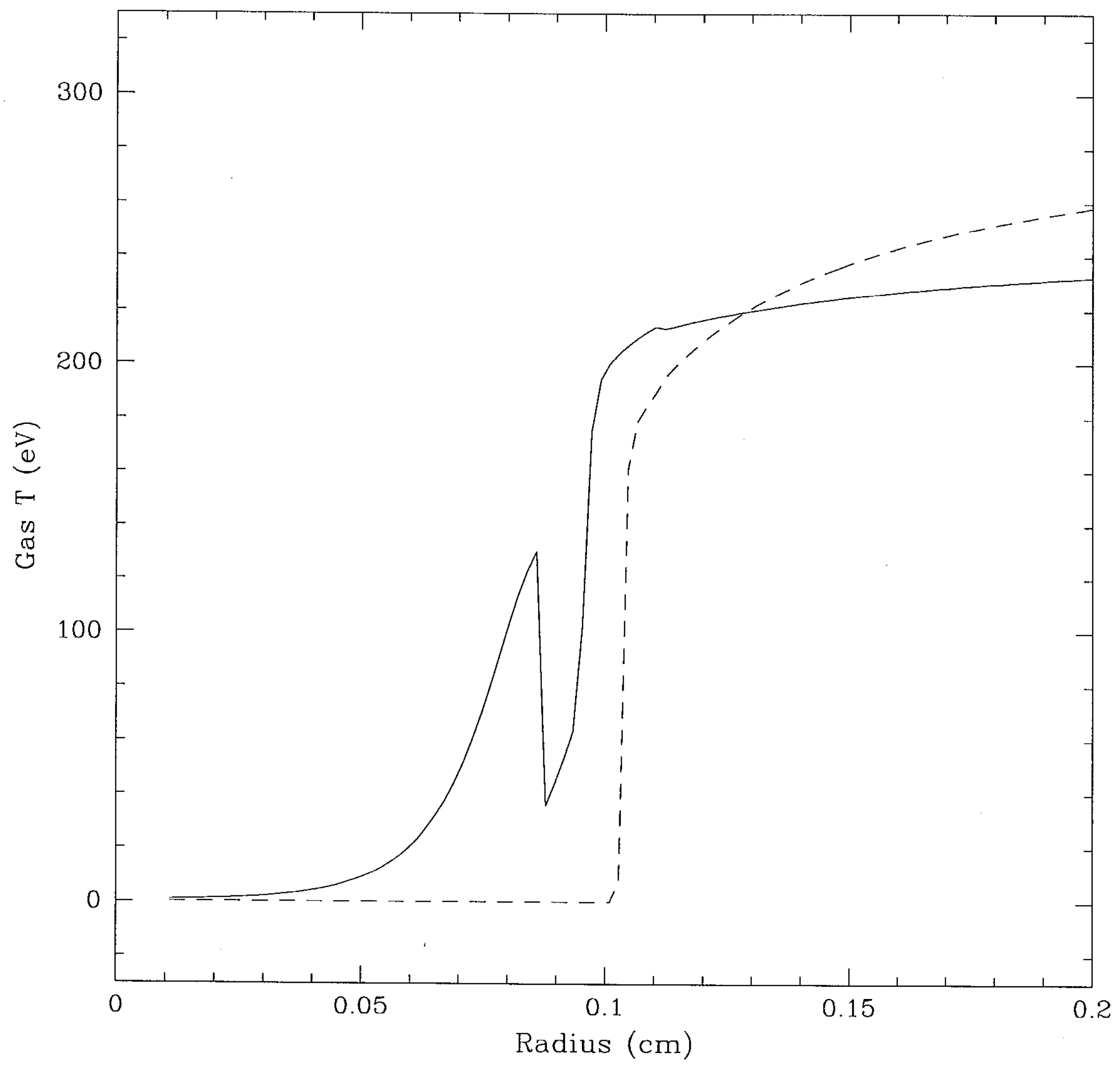

Fig. 6.— Diffusion with coupling: comparison of the gas temperature from the PF with C_{rad} = 0.005 (solid line) and 0.5 (dashed line).

evolution time.

To further investigate this behavior we considered controlling the timestep evolution by two other means. The first was to replace the radiation Courant condition with a fractional change restriction like that used on the matter temperature. The second was to use such a restriction on the *net* change in E, after both the diffusion and source/sink steps are completed. The former approach led, for the model considered, to timesteps which asymptoted to $\sim 10^{-19}$ seconds; the latter to timesteps which grew *extremely* slowly to 10^{-16} seconds but then dropped by an order of magnitude and leveled off. This behavior vividly underscores the extremely short evolutionary timescales for two different contributions to E (as contrasted with the much longer timescale for e), which are handled simultaneously and self-consistently in the implicit algorithm. At such timesteps, prospects for computing 3-D radiation hydrodynamic simulations with the explicit scheme are nil. Whether the potential for improved solutions via an alternate splitting scheme exists and should be pursued will be discussed further in §7.

5.2. Tests in Moving Media (ICF Test)

Because of the profound timestep limitations placed on the explicit algorithm, a direction comparison of the implicit and explicit algorithms on the full ICF problem at comparable accuracy is not currently possible. In §6 we will perform timing tests on the full ICF problem with the explicit machinery to estimate approximately how much the explicit algorithm would cost if it could be made accurate. These results will have direct bearing on the discussion of future work in §7. We will conclude our discussion of accuracy by documenting the timestep restrictions needed for reasonable accuracy in the implicit calculation of the ICF problem.

Of paramount importance in true ICF simulations is the temporal behavior of the central density, gas temperature, and radiation temperature. In the interest of economy, we wish to performance accuracy estimates using as few angular zones as possible, given that our version of the problem is spherically symmetric. We have found that for a given radial resolution (256 zones), the temporal behavior of the physical variables at the capsule's center is the same for angular (θ, ϕ) resolutions of (16,16), (32,32), (64,64), and (128,128). This result is likely due in large measure to the extreme focusing effect experienced as material converges on the center. Our comparisons of results with different timesteps are thus based upon a set of runs using the (16,16) angular grid. In the scaling studies presented in §6, grid sizes as large as (512,256,256) are used.

Hydrodynamic motions in both real ICF tests and our simplified version arise from ablation-driven implosion. Radiation from the hot (initially 300 eV) helium source region diffuses into the cold (initially 0.025 eV) carbon foam layer, raising the temperature and thus generating vigorous outward expansion. Due to momentum conservation, the region interior to the ablated material recoils inward, establishing implosive motion which ultimately elevates the central density by orders of magnitude (roughly 3 in our problem) over the starting value. As indicated in figure 7, a nascent ablation front is apparent by $t = 10^{-11}$ seconds. The velocities at the ablation and recoil fronts grow with time, with the maximum compression occuring at $t = 3.43 \times 10^{-9}$ seconds. The final profile shown in figure 7 corresponds to $t = 6.0 \times 10^{-9}$ seconds, well into the post-rebound phase of evolution.

Figures 8, 9, and 10 show the evolution of the central density, gas temperature, and radiation temperature, respectively, with time. The gas and radiation temperature plots both display unmistakeable signs of preheating in the core, beginning at a time of 1.27×10^{-9} seconds for the radiation temperature and 1.38×10^{-9} seconds for the gas temperature. This heating preceeds the main hydrodynamic compression, which begins at at time of 1.43×10^{-9} seconds (as evidenced by the density) and peaks at 3.43×10^{-9} seconds. In these results, the converged solutions were obtained with timestep restriction tolerances on both the radiation and gas temperatures of 0.01, and the acceptable solutions were obtained with tolerances of 0.10. The scaling and timing tests presented in §6 use this latter value.

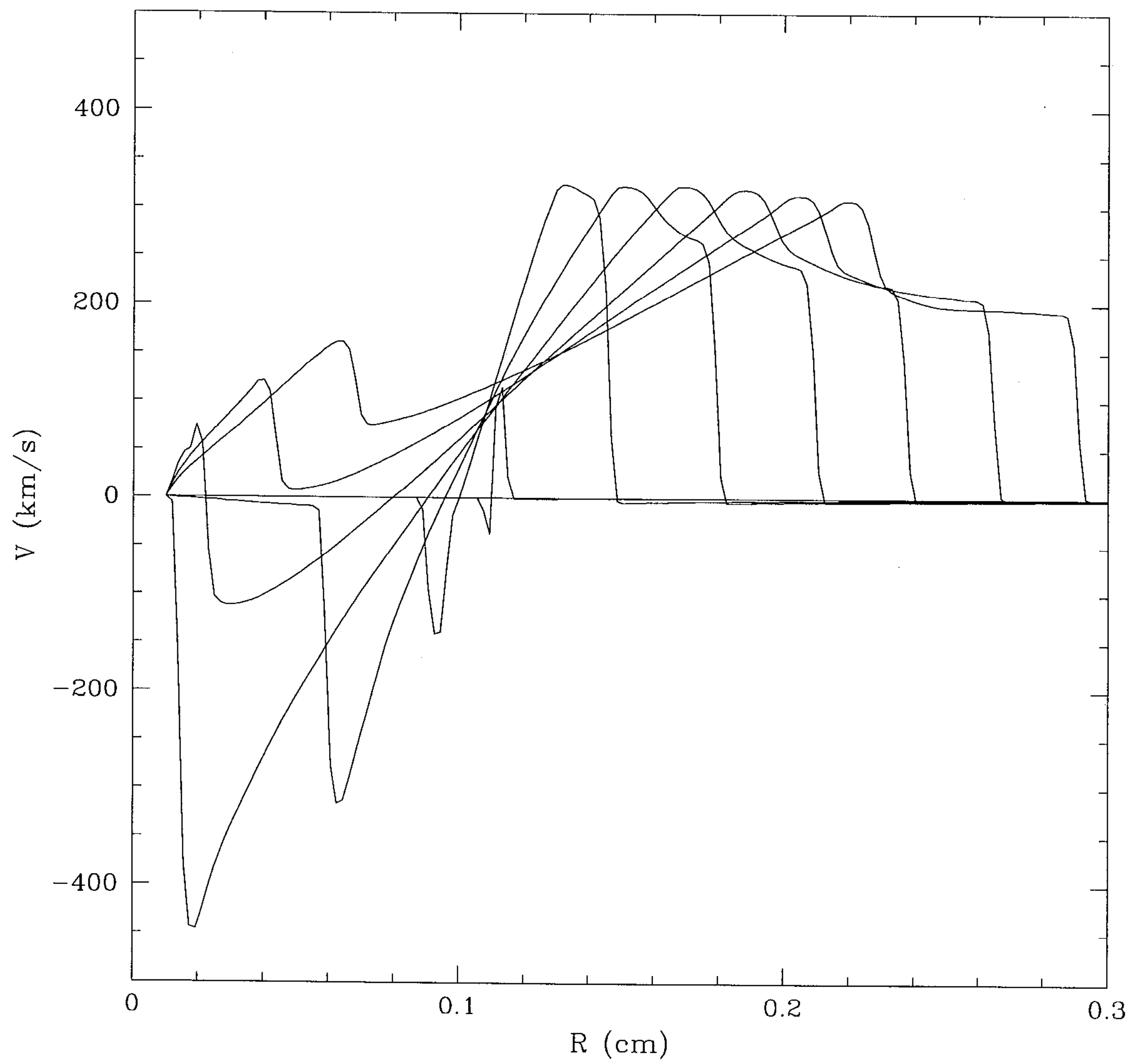

Fig. 7.— Implicit ICF capsule: radial velocity profiles at t = 10^{-11}, 10^{-10}, and (1,2,3,4,5,6)$\times 10^{-9}$ sec.

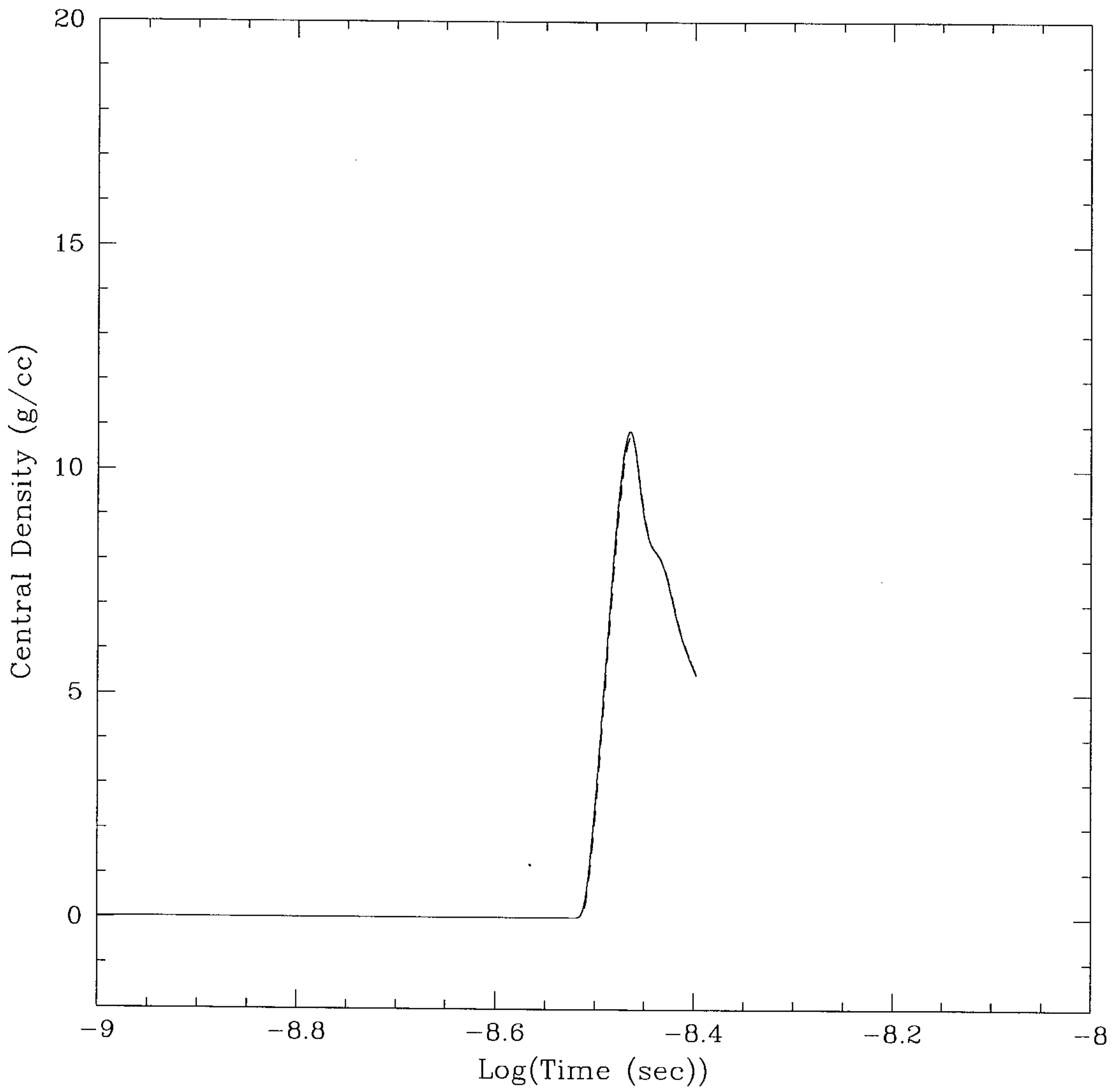

Fig. 8.— Implicit ICF capsule: evolution of the central density for tolerances of 0.01 (solid line) and 0.10 (dashed line).

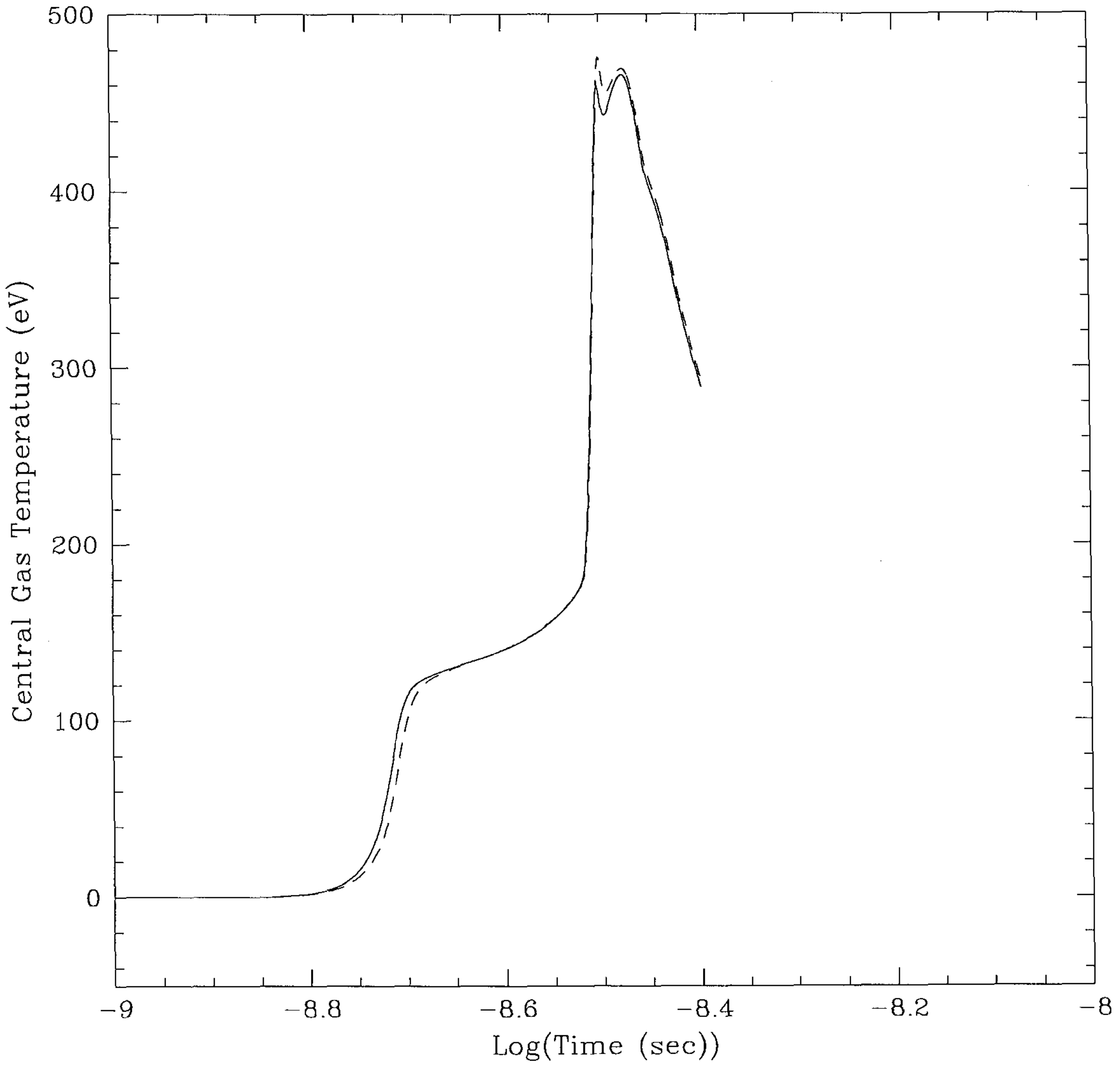

Fig. 9.— Implicit ICF capsule: evolution of the central gas temperature for tolerances of 0.01 (solid line) and 0.10 (dashed line).

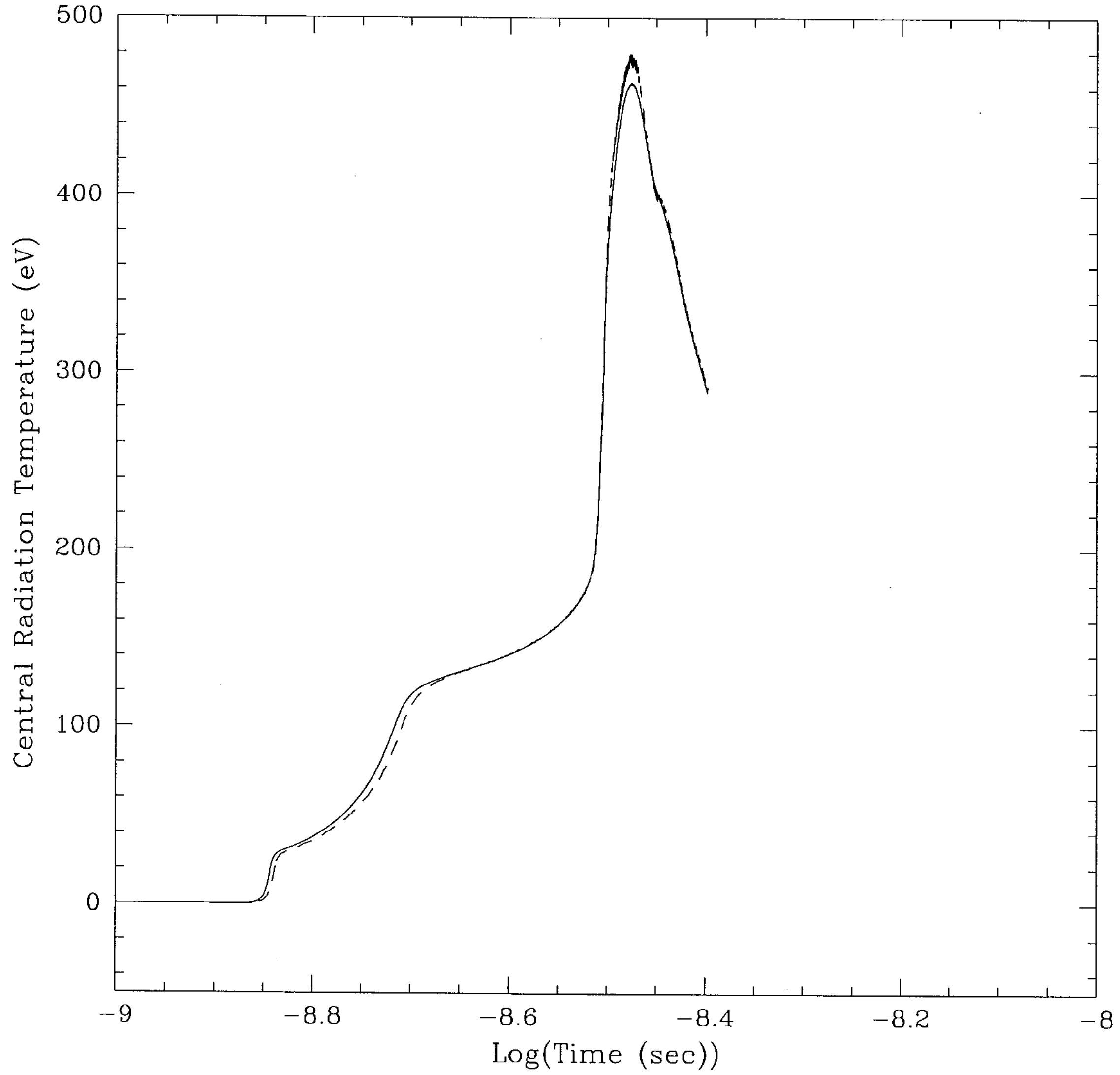

Fig. 10.— Implicit ICF capsule: evolution of the central radiation temperature for tolerances of 0.01 (solid line) and 0.10 (dashed line). At the temperature maximum, the upper curve is the dashed line.

6. Scaling Studies

Having established appropriate accuracy criteria for various tests of interest, we proceed to the central question of this report: scalability. In the context of iterative algorithms on parallel computers, the term "scalable" may have at least two distinct and equally important meanings. In the case of iterative linear solvers, "scalable" is frequently used to describe methods with iteration counts that have minimal dependence on problem size. In this context, multigrid methods are highly scalable and diagonally preconditioned CG methods scale very poorly. This issue has been explored in detail in Baldwin *et. al* (1998). This report focuses primarily on the complimentary view of scalability: the resilience of an algorithm's performance when distributed across an increasing number of processors. We mention in passing, however, that one of the chief incentives for investigating the PF algorithm is that it represents a limiting case of scalability of the first kind.

In a time-evolution calculation both types of scalability are critically important, and the two may interact in a manner that is often ignored in mathematical estimates of the operation count of a matrix solver. Specifically, consider diagonally preconditioned CG vs. MG. In the limit of small timesteps, the matrix to be solved is diagonally dominant, and a solution may be returned in a single iteration. As discussed in Baldwin *et. al* (1998), the single-node performance of the CG solver will surpass that of the MG solver owing to a substantially smaller CPU cost per iteration. However, as the timestep becomes larger, the matrix becomes less diagonally dominant. In this case, the iteration count for diagonally preconditioned CG can become very large, whereas the iteration count for MG remains low, typically less than 10. In this case, the cost of the MG algorithm can be significantly lower than that of the CG solver. In a hydrodynamic simulation, timesteps typically start at very low values, grow with time, and may vary by orders of magnitude during the dynamically interesting phase of a simulation. Thus it may easily be the case that the simulation will generate matrices, at different times, which may best be attacked by quite disparate methods. Indeed, this possibility has motivated us to develop a version of ZEUS-MP where the implicit solver can switch from CG to MG when the iteration count of CG becomes significant.

Convolved with the issue of CPU cost vs. evolution time is the question of how well an algorithm lends itself to multiprocessor operation. Preliminary scaling (with regard to node number) tests of MGMPI on simple tests (available on the WWW; see §1) suggests that resiliency of MG to node count may be problematic. Until we have an implementation of MG that can demonstrably handle the unusual numerical challenges of the ICF test, stronger judgement will have to be withheld. However, we will see below that our CG algorithm scales nearly as well as the PF algorithm with regard to node number, for two fundamentally different types of scaling tests. Additionally, the real possibility of a substantial reduction of the

nodes	CPU sec	zone-cycle/sec	GFLOP	speedup	parallel efficiency (%)
8	110.1	2.38×10^5	0.238	1.00	–
16	112.9	4.64×10^5	0.464	1.95	98
32	113.5	9.24×10^5	0.924	3.88	97
64	111.6	1.88×10^6	1.88	7.90	99
128	112.5	3.73×10^6	3.73	15.7	98
256	112.7	7.44×10^6	7.44	31.3	98
512	117.5	1.43×10^7	14.3	60.0	94
1024	117.8	2.85×10^7	28.5	120	94

Table 1: Pure Diffusion Scaling With CG: 32^3 zones/node

interation count with effective preconditioning (see §7) holds great potential for developing an algorithm that is highly scalable by both definitions.

All scaling runs presented here were computed on the ASCI Red machine at Sandia National Laboratory. The jobs were run in batch mode in either of two queues (llnl.day, llnl) from late September to late October. In all tests the code was compiled using "-O2 -Mnoframe -Munroll" as compile-line options, and "-Knoieee" on the link line. This choice gave the best performance among those suggested by Paul Work at LLNL. In the second set of scaling tests, where the total grid size is held fixed, the number of zones per node varied from a maximum of 64x64x32 (32 nodes) down to 16x16x16 (1024 nodes). In the first set of tests, where the number of zones/node is fixed, we chose an intermediate value of 32^3 zones/node. This gives a range of total grid sizes ranging from 256x32x32 at 8 nodes up to 512x256x256 at 1024 nodes.

6.1. Scaled Total Work

The first class of scaling tests involves sequences of runs where the number of zones owned by a node remains constant; therefore the total amount of work scales linearly with the number of nodes used. Because the physical size of the domain we model is held constant, the grid resolution along a given coordinate axis increases as the number of nodes spanning that axis increases. This means that the numerical character of the model changes in three potentially important ways. First, increasing the resolution lowers the radiation Courant time used in explicit tests, therefore increasing the number of timesteps needed to arrive at a fixed evolutionary point. Secondly, dynamic changes in the medium are better resolved,

nodes	CPU sec	zone-cycle/sec	GFLOP	speedup	parallel efficiency (%)
8	84.74	3.09×10^5	0.240	1.00	–
16	84.78	6.18×10^5	0.480	2.00	100
32	84.76	1.24×10^6	0.963	4.00	100
64	84.83	2.47×10^6	1.92	8.00	100
128	84.92	4.94×10^6	3.84	16.0	100
256	85.11	9.86×10^6	7.66	31.9	100
512	84.96	1.97×10^7	15.3	63.9	100
1024	86.08	3.98×10^7	30.9	126	98

Table 2: Pure Diffusion Scaling With PF: 32^3 zones/node

which means that a given tolerance on allowed changes per timestep of the various physical quantities can have a larger effect than in lower resolution runs. Third, the required number of iterations per timestep for the CG solver increases with zone number; thus the workload per timestep increases independent of other effects. Because the purpose of this set of scaling tests is to gauge the scalability with regard to node number, all scaled-work tests are begun with very low initial values of the timestep and run for 100 models, which covers a portion of evolution time so short that the variable numerical character of different resolutions does not manifest itself. The implicit runs use only 1 CG iteration per timestep in nearly all cases; in the exceptions the increase is too small (a fraction of an iteration in the average value) to be significant in computing speedup.

Tables 1 and 2, combined with figure 11, provide a direct comparison of the implicit and explicit diffusion solvers with the rest of the code machinery turned off. In the implicit test, the number of CG iterations per timestep was unity except for the 1024 node case, where it is 1.27. Such a small increase affects the total CPU only at the level of a few percent, which is comparable to the variations observed due to system effects. Therefore we may properly gauge code performance by multiplying the total number of zones by the number of timesteps used (100), and dividing by the "CPU sec", which is the total time used on the master thread. Because the load is balanced evenly across processors, thread times never differ by more than a few percent (when the system is operating normally). Therefore we define "speedup" as the ratio of the zone-cycles/sec to that value obtained with the baseline run, which in the scaled-work tests is performed on 8 processors. The speedup so defined is presented in the next-to-last column in tables 1, 2, and 3. With this definition, perfect scaling would be represented by a speedup equal to the number of nodes divided by 8. The percentage of that value that is actually achieved is given in the final column as "parallel

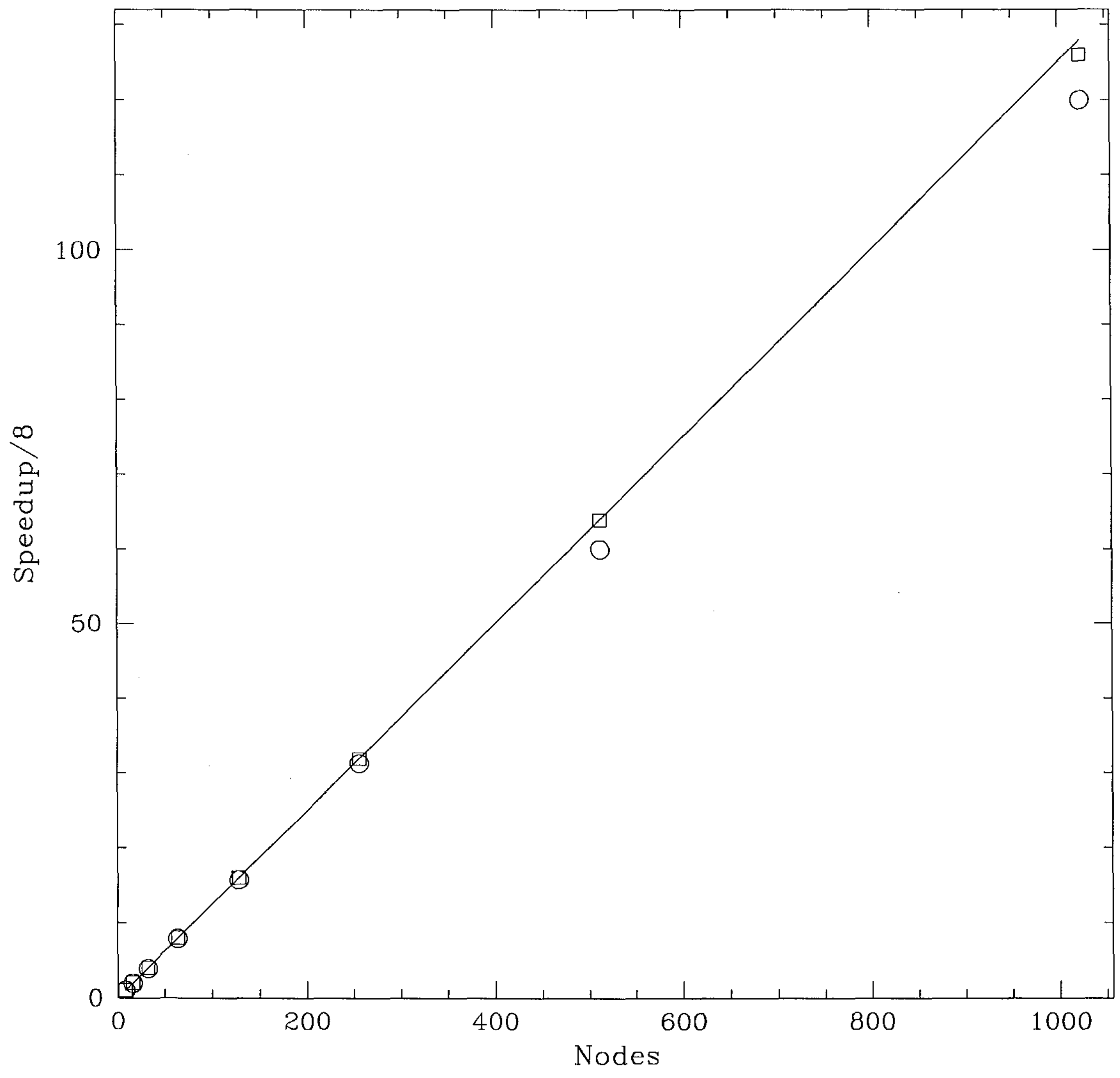

Fig. 11.— Pure diffusion on a scaled grid: speedup vs. node number for the CG solver (open circles) and the PF solver (open squares) on a scaled grid with 32^3 zones/node. The solid line is the theoretical perfect scaling limit.

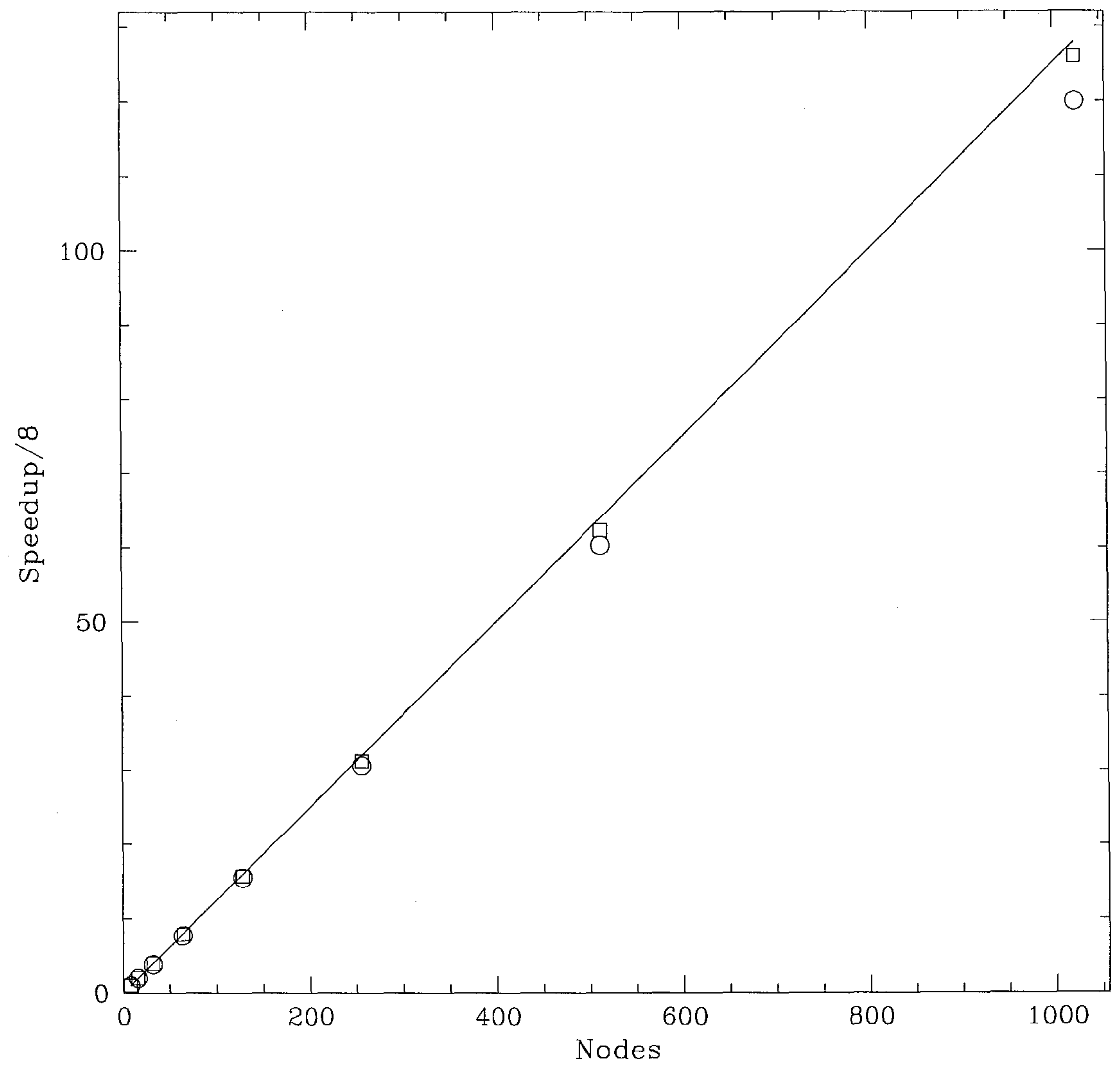

Fig. 12.— Radiation hydro on a scaled grid: speedup vs. node number for full radiation hydro (ICF capsule) with the CG implicit solver (open circles) and the PF solver (open squares) on a scaled grid with 32^3 zones/node. The solid line is the theoretical perfect scaling limit.

nodes	CPU sec	zone-cycle/sec	GFLOP	speedup	parallel efficiency (%)
8	236.0	1.11×10^5	0.267	1.00	–
16	239.1	2.19×10^5	0.528	1.97	99
32	246.1	4.26×10^5	1.03	3.84	96
64	245.5	8.54×10^5	2.96	7.69	96
128	243.8	1.72×10^6	4.14	15.5	97
256	246.9	3.40×10^6	8.19	30.6	96
512	250.6	6.69×10^6	16.1	60.3	94
1024	253.1	1.33×10^7	32.1	120	94

Table 3: Radiation Hydro Scaling With CG: 32^3 zones/node

nodes	CPU sec	zone-cycle/sec	GFLOP	speedup	parallel efficiency (%)
8	206.4	1.27×10^5	0.308	1.00	–
16	206.9	2.53×10^5	0.614	2.00	100
32	208.5	5.03×10^5	1.22	3.96	99
64	208.9	1.00×10^6	2.43	7.90	99
128	209.8	2.00×10^6	4.86	15.7	98
256	212.0	3.96×10^6	9.62	31.2	98
512	212.2	7.91×10^6	19.2	62.3	97
1024	210.4	1.59×10^7	38.6	126	98

Table 4: Radiation Hydro Scaling With PF: 32^3 zones/node

efficiency."

This measure of scalability yields results that virtually duplicate the theoretical limit for the explicit solver, and closely approach it for the implicit case. Furthermore, solving the full set of radiation hydrodynamic equations has very little effect on the scaling relative to the pure diffusion case. This is illustrated by comparing tables 1 and 3, or tables 2 and 4.

6.2. Fixed Total Work

Taken alone, the results of the scaled-work tests would seem to indicate that ZEUS-MP is an almost perfectly scalable code (with regard to node number) regardless of whether the explicit PF or implicit CG algorithm is used for diffusion. However, our second class of scaling tests yields a markedly different measure of the scalability. In scaled-work tests, both the computational load and communication overhead on each node are constant as the node number is increased. In fixed-work tests, however, both of these quantities decrease with increasing node number, but the ratio of communication overhead to computational load increases with node number. While we will quantify this statement below, the truth of it may be realized simply by remembering that the ratio of communication to computation varies by analogy to the ratio of surface area to volume of a regular solid: the smaller the characteristic size, the larger the ratio. In our case, communication costs are dependent upon the "surface area" of the grid domain on one node, whereas the computation required depends upon the grid volume.

We first illustrate the effects of varying the communication to computation ratio (CCR) through a comparison of the implicit and explicit algorithms on a fixed grid. Tables 5 and 6 list CPU times for calculations of pure diffusion on a fixed grid of 256x128^2. In contrast to the scaled-work tests, the fixed-work tests cover a significantly longer period of evolution time, which is the same for all runs of a given class of test. The pure diffusion runs terminate at t = 10^{-10} sec. While this is only a tenth of the time used in the (1-D) accuracy studies, we are constrained by the time required by the slowest (32-node) jobs, and the maximum available in readily accessible batch queues (14 hours). As mentioned previously, the number of zones/node varies from 64x64x32 on the 32-node runs down to 16^3 on the 1024-node runs.

Because the total work load is fixed, we may compute speedup by taking the ratio of CPU times between the run of interest and the baseline, which in these tests involve 32 nodes. We must acknowledge that comparison of the speedup relative to 32 nodes is unlikely to equal that which would result from comparison relative to a hypothetical 1-node baseline, but our minimum node number is determined by the memory capacity of an individual processor and

nodes	CPU sec	speedup	parallel efficiency (%)
32	49896	1.00	–
64	25239	1.98	99
128	14347	3.48	87
256	7989	6.25	78
512	4557	10.9	68
1024	2839	17.6	55

Table 5: Pure Diffusion Scaling With CG: 256x128x128 Grid

the memory needs of the code. Nonetheless, our comparisons within this class of test are made relative to a common baseline and therefore self-consistent. With our speedup defined as described, we may define a theoretical maximum from an N^{-1} dependence of CPU time on N, the node number, normalized to the 32-node baseline. Parallel efficiency is thus the percentage of that speedup actually achieved.

The speedups shown in tables 5 and 6 clearly demonstrate the effect of an increasing CCR as the node number increases. To lend a quantitative measure to the CCR, let us consider the amount of data exchanged between a pair of nodes during an MPI send/receive operation. In a typical message exchange, two layers of ghost zones are exchanged. A data slice in an MPI message has an "area" which is the product of the array dimensions defining the relevent grid face. ZEUS-MP arrays are 5 elements longer than the physical length, owing to the two ghost zones on each grid boundary, and an extra element which can accommodate special boundary conditions for the hydrodynamic advection scheme. The data slices sent are thus two layers with an area determined by the full array lengths. To count all the exchanged data points, an additional factor of two must be included as information is both sent and received. Careful counting of exchanged data points for a given nodal domain, divided by the number of the physical data points owned by a node, allows us to compute a ratio of array elements which are "communicated" to those which are "operated upon." We use this ratio to define the CCR.

As defined, the CCR ranges from 0.28 on the 32-node runs (64^2x32 zones/node) up to 1.27 for the 1024-node runs (16^3 zones/node). In scaled-work tests, we found that 16^3 zone/node runs scaled with node number equally as well as the 32^3 zone/node runs showed in the previous section. The reason for this is that the CCR is constant (0.48 and 1.27 for 32^3 and 16^3, respectively) on each node in scaled-work tests. Thus a calculation may possess a large amount of computational overhead, but if the overhead remains constant in a series

nodes	CPU sec	speedup	parallel efficiency (%)
32	944.1	1.00	–
64	483.3	1.95	98
128	255.1	3.70	93
256	136.8	6.90	86
512	73.57	12.8	80
1024	48.33	19.5	61

Table 6: Pure Diffusion Scaling With PF: 256x128x128 Grid

of scaling tests, then its effects are masked. In a sense, then, scaled-work studies represent a necessary but not sufficient test of the code+computer marriage: excellent performance on scaled-work tests is required for any hope of scalability in fixed-work tests, but does not guarantee it.

The true impact of the CCR will, of course, depend on the latency of the computer's network; i.e. the actual time required to pass a given amount of data across the network hardware. Rather than quote theoretical numbers for the Red machine, we present in table 7 lists of the time spent performing communication tasks in the CG solver vs. the total time spent in the CG subroutine call. Measuring true communication times in an algorithm where much of the communication cost is (in theory) hidden under computation with use of non-blocking sends and receives is a non-trivial exercise. We have chosen to measure and tabulate two communication related quantities. In the CG algorithm (as in the rest of the code), required sends and receives are posted, and then computation is performed. After the computation steps are complete, we initiate an MPI_WAITALL command to ensure that the non-blocking sends and receives have completed before proceeding. We define "send/receive" time as the total time spent waiting for a successful return from an MPI_WAITALL function call. This may be interpreted as the amount of send/receive communication time which was not buried inside the computation time. Our second time measure is "synchronization" time, which is the amount of time spent in MPI global reduction operations, which in the CG algorithm are MPI_SUM operations used for computing inner products. The sum of the send/receive (t_{sr}) and synchronization (t_{syn}) times is tabulated as t_{com}, the net communication overhead. This is divided by the total time spent in the CG routine (t_{rout}), and expressed as a percentage in the final column of table 7.

The timings presented in table 7 are sums for 100 calls to the CG solver, thus random

nodes	zones/node	t_{rout}	t_{sr}	t_{syn}	t_{com}	t_{rout}/t_{com}
32	64x64x32	55.0	1.12	0.483	1.60	2.9%
64	32x64x32	32.0	1.20	0.537	1.74	5.4%
128	32x32x32	17.1	0.74	0.379	1.12	6.6%
256	32x16x32	10.0	0.60	0.366	0.96	9.6%
512	32x16x16	6.38	0.63	0.361	0.99	15%
1024	16x16x16	4.77	0.44	0.425	0.87	18%

Table 7: Net communication overhead in the CG algorithm during the ICF capsule test for a fixed grid of 256x128x128. 100 timesteps were performed.

fluctuations due to system effects should be well absorbed. The test problem used was the full ICF calculation with the CG solver. The final column illustrates the trend consistent with our previous exercise of data point counting, and shows that the actual times involved can indeed be significant. This effect is entirely lost in scaled-work tests.

With these timing results in mind, the scaling results from both scaled and fixed work tests may be well understood. Both the pure diffusion and radiation hydrodynamic tests (tables 8 and 9) continue to show speedup even at 1024 nodes with only 16^3 zones/node. That the speedup curves (see also figs. 13-14) has not turned over by this point testifies both to the scalability of the routines and that of the hardware network. We see that the parallel efficiency at large node number is reduced when the implicit algorithm is chosen over the explicit algorithm, but such a result is consistent with the fact that the implicit solver is much more communication intensive than its explicit counterpart. As in the scaled-work studies, the addition of hydrodynamics (e.g. compare table 5 with 8) does little to significantly affect the scaling.

Given the results in §5 concerning accuracy, the presence in this section of "ICF" tests with the explicit algorithm deserves comment. As has been mentioned previously and will be elaborated upon in §7, the chief source of inaccuracy in the explicit scheme lies not in the PF algorithm but in the operator-splitting method used to marry the PF with radiation sources/sinks and matter coupling. A reasonable question to ask therefore is whether a revision of the operator-splitting and coupling schemes is a worthy pursuit in future work. Revised schemes are highly unlikely to be less expensive than the current one, therefore it is useful to compare the current explicit scheme to the implicit one and ask whether a revised explicit algorithm would be competitive with an optimized implicit solver. The answer to that question hinges upon both scaling behavior and upon estimates of the CPU time to a completed solution. Because the implicit algorithm leaves considerable room for

nodes	CPU sec	speedup	parallel efficiency (%)
32	35710	1.00	-
64	19341	1.94	97
128	11029	3.40	85
256	6127	6.12	77
512	3647	10.3	64
1024	2126	17.6	55

Table 8: ICF Capsule Scaling With CG: 256x128x128 Grid

improvement with the use of (already known) preconditioners that can sharply reduce the required number of CG iterations, this scalability comparison and the timing comparisons to follow in §7 are interesting and important.

Tables 8 and 9 illustrate relative scaling behavior which is strongly similar to that of the diffusion solvers acting in isolation. The evolution time considered for the ICF test is the same as that of the pure-diffusion tests: 10^{-10} sec. This represents only 1/40 of the time needed for a complete hydrodynamic evolution, but it is certainly sufficient to gauge the scalability of the algorithms. Over this evolutionary time, the CG solver averaged roughly 170 iterations per timestep (this value nearly doubles by 10^{-9} sec), and thus the CPU cost is completely dominated by the CG solver. As seen previously, we still see speedup at 1024 nodes, when only 16^3 zones are owned by each node, but the parallel efficiency of the explicit solver is noticeably higher than that of the implicit solver. During the bulk of the evolution, the timestep chosen in the explicit tests was limited by the radiation Courant number (we chose a value of 5.0 for these tests), whereas in the implicit runs it is limited by the tolerances on radiation and gas energy fractional changes (0.1). In neither case, nor in the implicit runs which cover the entire physical evolution (§5), did the hydrodynamic Courant time play a significant role.

nodes	CPU sec	speedup	parallel efficiency (%)
32	6238	1.00	–
64	3326	1.88	94
128	1710	3.65	91
256	911.9	6.84	86
512	490.9	12.7	79
1024	276.5	22.6	71

Table 9: ICF Capsule Scaling With PF: 256x128x128 Grid

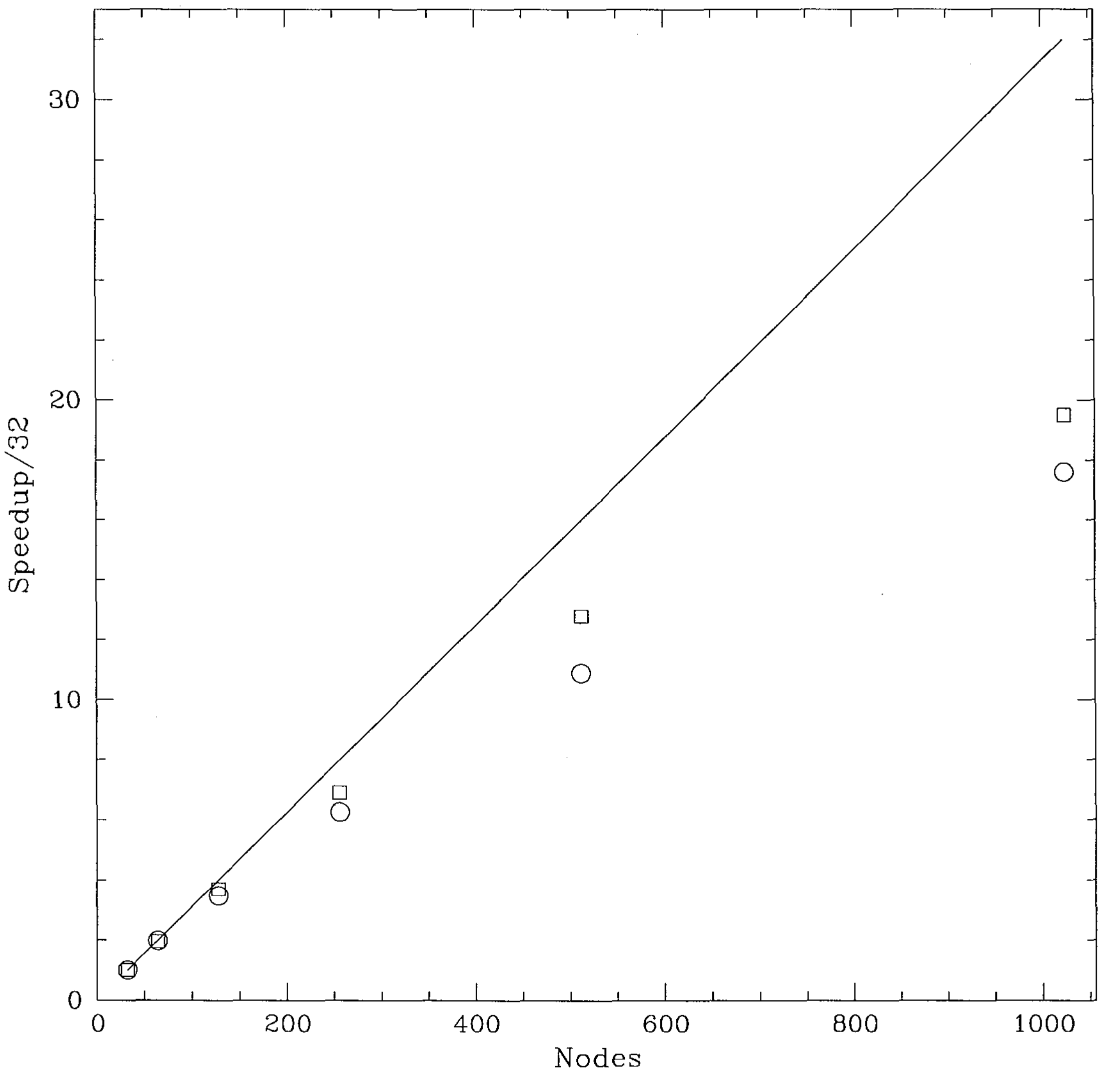

Fig. 13.— Pure diffusion on a fixed grid: speedup vs. node number for the CG solver (open circles) and the PF solver (open squares) on a fixed grid of 256x128x128. The solid line is the theoretical perfect scaling limit.

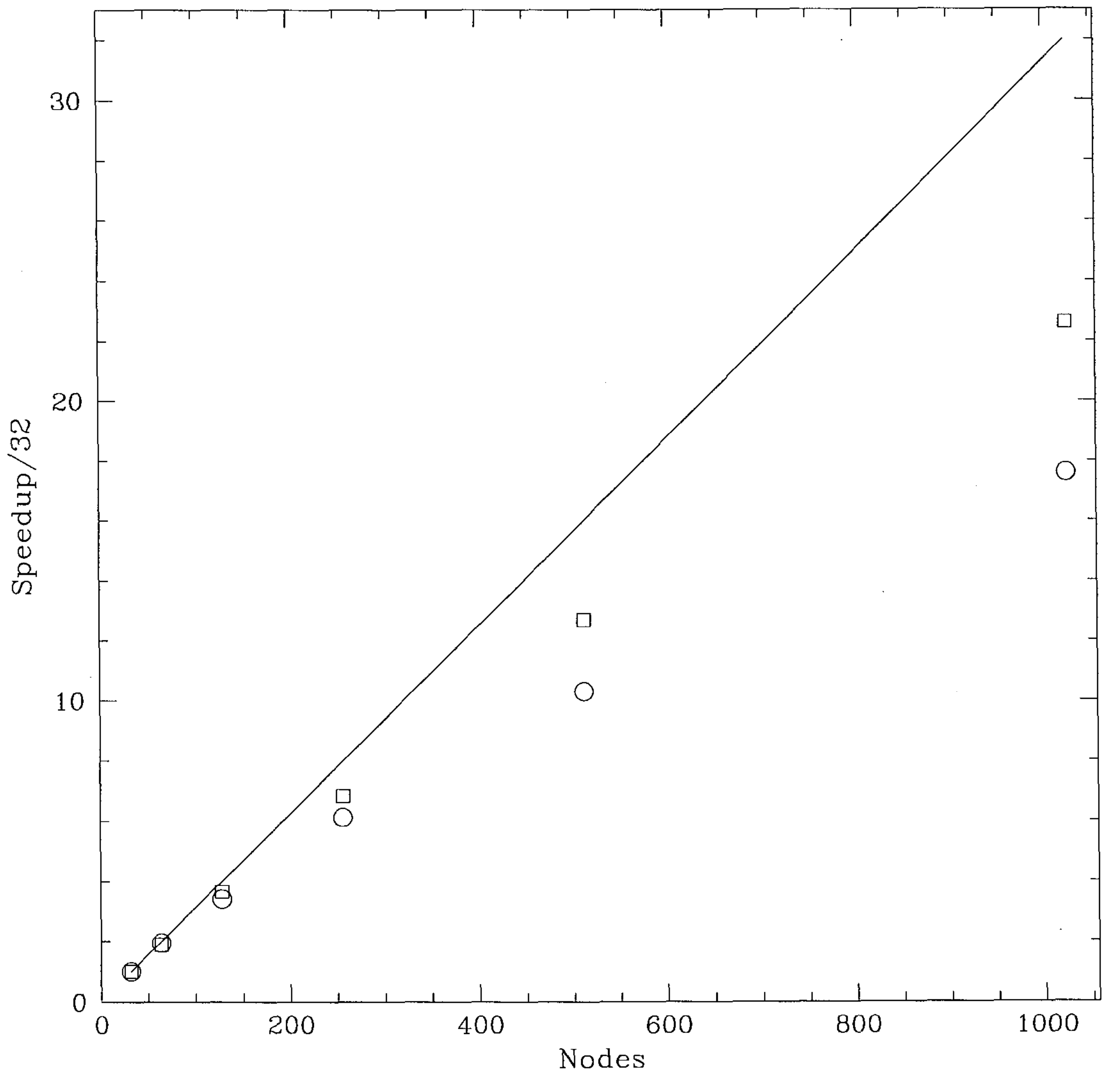

Fig. 14.— ICF capsule on a fixed grid: speedup vs. node number for the CG solver (open circles) and the PF solver (open squares) on a fixed grid of 256x128x128. The solid line is the theoretical perfect scaling limit.

7. Summary and Discussion

7.1. Current Results

The original purpose of this investigation was to compare the relative merits of the unconditionally stable, explicit Product Formula (PF) algorithm to those of an implicit approach using a preconditioned conjugate gradient linear equation solver. The aim was to compare scalability and time to solution at comparable accuracy for a specific test problem. The PF algorithm possesses several features which motivated our interest, namely: reduced communication and memory requirements, and lower operation count per timestep relative to the implicit approach. The relatively low cost per timestep, plus the possibility of enhanced scalability due to decreased communication demands, were the primary features piquing our interest in this approach. On the other hand, unconditional stability does not guarantee unconditional accuracy; thus the explicit evolution may still be bounded by some maximum timestep value which is not present in the implicit approach. Therefore, the number of timesteps needed for a complete solution, as well as the cost per timestep, must be considered. Implicit methods have generally been chosen for radiation transport over traditional (radiation Courant limited) explicit methods for this very reason.

To explore these issues, we have developed ZEUS-MP, a 3-D radiation hydrodynamics code written for use on massively parallel computers with the Message Passing Interface (MPI) standard. We then challenged the algorithms within ZEUS-MP by constructing a test problem inspired by simulations of inertial confinement fusion (ICF). Our code does not treat nuclear burning, multi-species fluids, non-planckian radiation fields, nor non-ideal effects in the material EOS; therefore our test problem may not be construed as a real ICF simulation. Nonetheless, physical parameters have been chosen so that an ablation-driven implosion on the proper timescale may be realized. Our problem thus consists of radiation from a hot, optically thin source penetrating and ablating material on a cold, optically thick target, with subsequent inward radiation diffusion and material implosion. We therefore exercise our radiation diffusion solvers in a manner that should be meaningful to those in pursuit of more sophisticated problems.

Our head-to-head comparisons of the implicit and explicit solutions involve not only the performance of the PF and CG solvers, but of the operator splitting schemes in which they are embedded. The implicit method uses a two-stage scheme wherein the radiation energy equation is solved with diffusion coefficients, opacities, and source terms evaluated using the matter temperature from the old timestep. We then update the matter temperature using a linearization procedure analogous to that employed in the ARES code (Baldwin *et. al* 1998). In the explicit scheme, the PF is used to evaluate the diffusion term exclusively. The remainder of the radiation energy equation is evaluated in a second sub-step, and then the matter temperature is updated exactly as in the implicit procedure.

In §5, we examined studies to determine timestep restrictions needed to produce aysmptotically correct behavior of the physical variables, and looser restrictions which allowed solutions that lay within roughly 10% of the "converged" values. These "tolerable" solutions then served as the basis for scaling comparisons given in §6. ZEUS-MP uses a variety of hydrodynamic constraints on the timestep (Courant time, maximum changes in velocity, artificial viscosity, etc.), but these must be supplemented with additional controls when radiation is included. In the explicit approach, the radiation Courant time, multiplied by a user-specified factor, is the default control regarding the radiation energy density. The implicit method employs a user-specified limit on $\Delta E/E$, where ΔE is the change in the radiation energy density in a given zone resulting from the implicit update. Both solvers use a similar limit on $\Delta e/e$, where e is the gas energy density.

Tests involving either pure diffusion or diffusion with coupling were performed in one dimension, owing to the spherical symmetry of our problem. Tests including hydrodynamics were automatically run in three dimensions. All accuracy tests used 256 radial zones, but some of the scaling tests used as many as 512 radial zones. The number of angular zones varied from as few as 16^2 to as many as 256^2. In full radiation hydrodynamic runs covering all (or nearly all) of the implosion phase, we found that the evolution of the central density, radiation temperature, and gas temperature were independent of the angular resolution, which again is a consequence of the spherical symmetry we imposed.

When using the implicit method, we found that converged solutions were achieved by choosing tolerances on $\Delta E/E$ and $\Delta e/e$ in the range 0.005 to 0.01. Tolerable solutions (as we have defined them) were achievable with these restrictions loosened to 0.1 for both parameters.

Regarding the explicit method: we found, for tests involving pure diffusion only, that asymptotically correct solutions were approached for $C_{rad} \leq 0.5$, but that tolerable solutions were retrievable for $C_{rad} \leq 10$. Unfortunately, we discovered that such values of C_{rad} were possible *only* for the case of pure diffusion; once matter-radiation coupling is included, solution accuracy is totally lost unless C_{rad} is made to lie below (for this problem) 0.005. This result is a direct consequence of the 3-stage operator splitting scheme, combined with the default method of controlling timesteps. In § 5, we noted that when reasonable tolerances on $\Delta E/E$ were used for timestep regulation, rather than C_{rad}, timesteps remained at extremely low values: on the order of 10^{-17} seconds. A reasonably accurate simulation of the ICF test with the explicit approach would thus require roughly 400 million timesteps.

While we have shown that the current explicit scheme is not usable for the ICF test, we have not determined whether an alternate splitting scheme would allow for productive use of the PF approach. However, we may pose the question as to whether such effort is even worth pursuing. To that end, we will continue by reviewing the scaling data we have generated for both solvers, and then by considering timing data for full-term runs of the ICF test with both solvers. We will show that both published and continuing work on advanced preconditioners – needed in the implicit linear solver – gives substantial hope for an implict solver which is more cost effective than a hypothetical accurate explicit scheme.

Our scaling tests were divided into two fundamental classes: tests where the total amount of work scaled linearly with the number of nodes, and tests where the total work was held fixed. As discussed at length in §6, scaled-work tests represent a necessary hurdle that a scalable code (and computer architecture) must master, but are not of themselves sufficient to demonstrate true scalability with regard to node number. The reason for this, as evidenced by the fixed-work tests, is that in scaled-work tests the amount of communication overhead experienced by a CPU is essentially fixed once the domain has been decomposed along all three coordinate axes; thus the effects of even a large communication to computation ratio (CCR) are well masked. Indeed, the scaled-work tests for both algorithms, whether in isolation or with hydrodynamics, show only slight departures from linear speedup through a node number of 1024, with no real indication of a major slowdown due to communication cost.

The fixed-work tests clearly highlight the effect of an increasing CCR as the node number grows, but even here the differences between the explicit and implicit solvers are not as large as might have been anticipated. In both pure-diffusion and radiation hydro tests, the speedup curves retained strongly positives slopes even when the number of zones per node had dropped to 16^3, and it is unlikely that a production calculation would use a per-node number substantially smaller than this. Therefore, with regard solely to scalability with respect to node number, both algorithms have acquitted themselves admirably.

7.2. Implicit vs. Explicit: Likely Prospects

Having collected and summarized our accuracy and scaling tests as they currently stand, we now examine ICF runs, with both solvers, that cover the entire implosion phase and terminate at the point of core rebound. The showcase of this discussion is a collection of timings presented in table 10. These timings pertain to a calculation of the full ICF test on a $256\text{x}128^2$ grid.

The first column of table 10 contains evolutionary time (all times are in seconds) of the simulation. The final time shown is 90% of the evolution covered covered in the low-resolution tests shown in §5. In order to run the ICF test to completion at high resolution with the implicit solver, it was necessary to run the job on an Origin2000 machine at NCSA, as none of the batch queues for janus (the Red machine) had sufficiently long time limits. Since we are concerned with time values on janus, we have done the following: the O2K run logged both evolution time and cumulative CPU time at each timestep. From these data, we determined that it required 28.6 times as much CPU time to evolve to $t = 3.6 \times 10^{-9}$ sec as it did to evolve to $t = 1.0 \times 10^{-10}$ sec. This ratio was then used to scale the implicit result from janus, which terminated at 10^{-10} sec, to a projected value for the full evolution time. The first number in column 3 is an actual janus timing, and the subsequent numbers are scaled estimates. The O2K results were run on 64 nodes of a 128-node machine. Because NCSA jobs are not over-scheduled with regard to node requests, there is no competition for node use from other jobs. Because of this the NCSA batch queues quite closely approximate true dedicated queues in terms of performance. Therefore we feel that using our advertised ratio to scale timing results from another machine leads to estimates which are reasonable enough to warrant the conclusions we will draw from them.

Column 2 contains estimates of the time which would be required for the current explicit scheme to obtain a comparably accurate solution to the ICF problem. These estimates are made with two pieces of information. The first is the cost per timestep of the explicit algorithm, which was determined by running the code on janus for a large number of timesteps, and dividing the CPU time used by the total number of timesteps. The second piece of information is an approximate value of the timestep required for an accurate solution. The value chosen was 10^{-17} seconds, and was taken from our discussion in §5. The CPU times for various evolution times are then estimated and shown in column 2.

Column 3 contains the actual and projected timings for the current implicit solver as it

t (evol)	Exp. CPU (current)	Imp. CPU (current)	Exp. CPU (hyp.)	Imp. CPU (hyp.)	<CG Iter> (current)
1.0×10^{-10}	3.5×10^{6}	2.1×10^{3}	2.8×10^{2}	2.1×10^{2}	170
1.0×10^{-9}	3.5×10^{7}	1.2×10^{4}	2.3×10^{3}	1.2×10^{3}	310
3.6×10^{-9}	1.3×10^{8}	6.0×10^{4}	8.1×10^{3}	6.0×10^{3}	350

Table 10: Approximate timings for accurate ICF solutions with current and hypothetical ("hyp.") PF and CG solvers. All times are in seconds.

performs on janus. We had fully intended to extend the runs to the evolution time considered in the accuracy tests (4.0×10^{-9} sec), but random and persistent system outages in the O2K platform prevented us from obtaining the full-length run. Nonetheless, 90% of the full run will allow us to discern the key points of this exercise. Column 4 contains timings for an explicit run performed fully on janus. While the solution was hopelessly inaccurate, we present these timings as lower limits to what may be expected from a "hypothetical" explicit scheme which can produce an accurate result. We do this under the reasonable assumption that such a solver is unlikely to be *less* expensive than the one currently implemented. This run was completed with a C_{rad} of 5.0, and a tolerance of 0.05 on the fractional change in the gas energy density.

Column 5 presents the most speculative numbers in the table. In a comparison of timings using CG with two different preconditioners, Baldwin *et. al* (1998) found that, for a spherically symmetric implosion problem in 2-D, CG with an ICT (Incomplete Cholesky with Thresholding) preconditioner returned a solution in less than one tenth the time required by CG with diagonal scaling (DS). Refering to table 12 of their paper, we see that their ICT+CG solver showed speedups of 1.16, 3.12, and 11.82 relative to DS+CG, for problem sizes of 900, 10^4, and 9×10^4 zones. The speedup ratio grows dramatically with problem size because for diagonally preconditioned CG, the average number of iterations required each timestep varies strongly with problem size, whereas the dependence shown by ICT+CG is much weaker. Our test problem involved 4.2×10^6 zones, which is nearly 47 times the size of the largest 2-D spherical implosion problem considered in the Baldwin paper. Guided by these results, we ask what level of speedup might be realizable in our implicit CG solver if an advanced preconditioner is substituted for diagonal preconditioning. This speedup is dependent upon both the reduction in the number of CG iterations and the relative cost of the preconditioning operation. Given that the Baldwin group achieved over a factor of 10 relative to diagonal preconditioning on problem sizes far smaller than ours, we regard a factor of 10 as a conservative estimate of the amount of speedup potentially realizable in our scheme. Unlike the ARES code, ZEUS-MP spends an amount of time doing hydro and supplemental functions which is very small compared to the CG solve once the number of iterations becomes much larger than unity. Therefore, a speedup in the CG solver relative to diagonal preconditioning is highly indicative of the speedup of the entire code. We thus display in column 5 rough predictions of the total CPU cost for the implicit solver with an "advanced" preconditioner. These estimates are merely those taken from column 3, with an order of magnitude reduction. To better appreciate the amount of room for improvement in our current DS+CG scheme, we include a running average of the total number of CG iterations per timestep required as a function of evolution time. These numbers are shown in column 6. Swesty (private communication) has found, in work being prepared for publication, that the number of CG iterations needed in large diffusion problems is of order 10 when preconditioners based upon approximate inverses are used.

Columns 3 and 4 of table 10 give timings with the implicit and explicit solvers as they are currently implemented in the code, with typical choices of timestep control parameters. While the results of the explicit run in column 4 were not usable, we identify the timings as representing a lower limit to the timings that would result from a hypothetical modified scheme (hence the "hyp." designation) that could yield accurate results with a C_{rad} of 5-10. Column 2 shows what would be roughly needed for an accurate solution with the existing explicit scheme, and demonstrates quite clearly that the prospects for ICF calculations with the explicit algorithm are hopeless as things currently stand. Column 3 gives representative timing values with the current implicit solver, and column 5 gives predicted estimates for what could be obtained with existing advanced preconditioners suitably adapted for parallel use. We emphasize that the current preconditioning employed is the least effective choice possible (apart from none). Furthermore, extensive work into parallel preconditioners has already resulted in at least two library packages (ParPre and PETSc) containing a variety of preconditioners designed for parallel use, thus there are already a variety of choices available for further inquiry and research. Our numbers, taken at face value, imply (1) that there exists considerable hope for an implicit solver that is far more economical than the diagonally preconditioned solver currently in place, and (2) it will be difficult, if not impossible, to produce an explicit routine that is dramatically superior to an advanced implicit solver. This latter statement would be refuted if either (1) the PF routine could be run at values $>> 10$, or (2) if advanced preconditioners in the CG algorithm in all cases destroyed the scalability (with regard to node number) of the solver. Even pure diffusion tests with the PF are limited to values of C_{rad} of 10 or less if accuracy is to be retained, so it is difficult to envision such a limitation being overcome regardless of the splitting scheme chosen to include sources, sinks, and matter coupling. Therefore we feel that the prospects for running the PF at high radiation Courant factors are virtually nil. The scalability of advanced preconditioners is perhaps the issue of greatest importance when possible choices are considered, but given that a minor industry has arisen in pursuit of preconditioners that are scalable both with regard to problem size and node number, we feel that the prospects for a high-performance implicit scheme are greater than those for an efficient scheme based upon the Product Formula, at least so far as problems similar in nature to the ICF capsule are concerned. We therefore suggest that future research efforts at treating problems of this sort will be most profitably directed toward implicit schemes with optimized iterative solvers.

7.3. Future Work

A. Discretization With Covariant Coefficients on a Staggered Mesh

Here we present a few simple relations which are used extensively in Appendices B and C. ZEUS-MP uses volume differencing and covariant metric coefficients in a scheme identical to that used in ZEUS-2D and ZEUS-3D. A full treatment of the scheme may be found in Stone & Norman (1992), and Stone, Mihalas, & Norman (1992), but we will present here only what is needed to decipher the formulae given in the following appendices.

ZEUS-MP uses a staggered mesh, with energies, densities, temperatures, and opacities defined at zone centers, and velocities, fluxes, and diffusion coefficients defined at zone interfaces. We thus refer to an "A" mesh (with quantities labeled with subscript "a") and a "B" mesh (with quantities labeled with subscript "b"). The A mesh is coincident with zone interfaces, while the B mesh is coincident with zone centers.

The equations in ZEUS-MP are cast as functions of covariant metric scale factors, g_1, g_2, g_{31}, and g_{32}, which have the following values:

$$g_1 = 1 \text{ (all coordinate systems)}, \tag{A1}$$

$$g_2 = \begin{cases} 1 & \text{(cartesian)} \\ 1 & \text{(cylindrical)} \\ r & \text{(spherical)}, \end{cases} \tag{A2}$$

$$g_{31} = \begin{cases} 1 & \text{(cartesian)} \\ r & \text{(cylindrical)} \\ r & \text{(spherical)}, \end{cases} \tag{A3}$$

and

$$g_{32} = \begin{cases} 1 & \text{(cartesian)} \\ 1 & \text{(cylindrical)} \\ \sin\theta & \text{(spherical)}. \end{cases} \tag{A4}$$

With these definitions, we may document the formulae for the gradient of a scalar, E, and the divergence of a vector, F:

$$(\nabla E) = \left(\frac{\partial E}{\partial x_1}, \frac{1}{g_2}\frac{\partial E}{\partial x_2}, \frac{1}{g_{31}g_{32}}\frac{\partial E}{\partial x_3}\right); \tag{A5}$$

$$\nabla \cdot F = \frac{1}{g_2 g_{31} g_{32}}\left[\frac{\partial}{\partial x_1}(g_2 g_{31} g_{32} F_1) + \frac{\partial}{\partial x_2}(g_{31} g_{32} F_2) + \frac{\partial}{\partial x_3}(g_2 F_3)\right]. \tag{A6}$$

Finite-difference expressions in ZEUS-MP are written as functions of volume differences. If we define Δx_1, Δx_2, and Δx_3 as differential length elements along the 3 respective coordinate axes, then one may show that corresponding differential volume elements may be written as follows:

$$\Delta V_1 = g_2 g_{31} \Delta x_1; \tag{A7}$$
$$\Delta V_2 = g_{32} \Delta x_2; \tag{A8}$$
$$\Delta V_3 = \Delta x_3. \tag{A9}$$

In appendix B, we will use the expressions given above in the formulae for the PF matrix elements, but in appendix C, we will write the final matrix element expressions in terms of actual array names used in ZEUS-MP. Doing so will avoid expressions which, when written in discrete form, become a blur of parenthesized zone indices, subscripts, and superscripts. Therefore we tabulate array quantities which represent the fundamental grid variables used in the following appendices:

Direction	g_2	g_{31}	g_{32}	Δx	ΔV
1	g2a(i)	g31a(i)	—	dx1a(i)	dvol1a(i)
2	—	—	g32a(j)	dx2a(j)	dvol2a(j)
3	—	—	—	dx3a(k)	dvol3a(k)

Table 11: Metric coefficients, zone widths, and differential volume elements for the "a" mesh. Corresponding elements for the "b" mesh are found by replacing "a" with "b" in the array names.

Direction	g_2^{-1}	g_{31}^{-1}	g_{32}^{-1}	Δx^{-1}	ΔV^{-1}
1	g2ai(i)	g31ai(i)	—	dx1ai(i)	dvol1ai(i)
2	—	—	g32ai(j)	dx2ai(j)	dvol2ai(j)
3	—	—	—	dx3ai(k)	dvol3ai(k)

Table 12: Inverse metric coefficients, zone widths, and differential volume elements for the "a" mesh. Corresponding elements for the "b" mesh are found by replacing "a" with "b" in the array names.

Tables 11 and 12 give, using the "a" mesh as an example, a listing of the most commonly used differential element expressions in the code, along with their corresponding array names. Table 12 shows the reciprocal values of the quantities given in table 11. A completely analogous set of quantities exists for the "b" mesh as well.

B. The Generalized Product Formula

In this appendix we present the full machinery for evaluating the radiation diffusion term via the Product Formula (PF). Recall that in this formalism we are solving the following equation,

$$\frac{dE}{dt} = \nabla \cdot (D\nabla E), \tag{B1}$$

which we recast in matrix form as

$$E_i(t+\Delta t) = \exp(\Lambda_{ij}\Delta t)E_j(t). \tag{B2}$$

As shown, equation B2 may be interpreted as the solution to a one-dimensional, discretized form of the diffusion equation. Following closely the procedure outlined in Graziani (1995), we use (B2) to solve the 3-D problem by performing sequences of 1-D sweeps along the three coordinate axes. The order of the sweeps is permuted from one timestep to the next in a manner identical to that employed in the hydrodynamic advection step. Our overall implementation of the PF is identical to that described in the Graziani paper, and the reader is strongly encouraged to reference that work for more background discussion. The differences in between our new algorithm and that outlined in the Graziani paper lie in the values of the matrix elements, which in ZEUS-MP are written using the covariant metric coefficients on the staggered mesh, and which in general need not produce a uniform grid. This feature breaks the matrix symmetry present in the case of uniform cartesian meshes, for which the Graziani algorithm was specifically designed. Rather than regurgitating the full discussion presented in the Graziani paper, this appendix makes use of the identical concepts and terms defined therein, and presents our exponentiated matrices (the heart of the PF method) as extensions of the simpler versions presented in the Graziani work.

We will illustrate the method by considering a 4-zone problem in one dimension. With minimal algebra the "div-grad" term may be represented as a matrix of the following form:

$$M = \begin{pmatrix} -\theta_l A + B & B & 0 & 0 \\ C & -(C+D) & D & 0 \\ 0 & E & -(E+F) & F \\ 0 & 0 & G & -(G+\theta_r H) \end{pmatrix}. \tag{B3}$$

In this general form, we see that the superdiagonal terms (B, D, F...) are not equal to their transpose-counterparts on the subdiagonal (C, E, G...), thus the matrix is not symmetric in the usual sense. This asymmetry arises because of nonuniform grid expressions buried in the matrix elements (but which will be detailed below). Additionally, we see that the main diagonal is expressible as a sum of the sub- and super-diagonal terms on the same row. Following the convention in Graziani (1995), we use θ_l and θ_r to represent the effect of a reflecting (closed) or non-reflecting (open) boundary condition on either side. In the case of closed B.C.'s $\theta_l(\theta_r) = 0$; otherwise $\theta_l(\theta_r) = 1$. Numerically the effect of this is to include contributions from "ghost-zone" values of either a sub-diagonal term (at the inner boundary) or super-diagonal term (at the outer boundary).

The original Graziani algorithm was designed to further include the effects of time-variable sources at either boundary, and are thus part of the boundary conditions. These effects may be included elegantly by augmenting our matrix M by an additional row (and column) for each boundary. Denoting our inner (outer) boundary source terms with $\Gamma_l(\Gamma_r)$, we may define an expanded matrix Λ, as follows:

$$\Lambda = \begin{pmatrix} \Gamma_l\theta_l & -(1-\theta_l)B & (1-\theta_l)B & 0 & 0 & 0 \\ \theta_l A & (-\theta_l A + B) & B & 0 & 0 & 0 \\ 0 & C & -(C+D) & D & 0 & 0 \\ 0 & 0 & E & -(E+F) & F & 0 \\ 0 & 0 & 0 & G & -(G+\theta_r H) & \theta_r H \\ 0 & 0 & 0 & (1-\theta_r)G & -(1-\theta_r)G & \Gamma_r\theta_r \end{pmatrix}. \tag{B4}$$

Inspecting (B4), we see that in the case of reflecting B.C.'s the boundary rows duplicate the rows immediately interior, which correspond to the first and last physical grid points. Simultaneously, the extreme left and right columns become null vectors. For open boundaries, the boundary rows represent evolution equations for the source terms, which or may not be zero.

The matrix given in (B4), when exponentiated, would in principle allow the PF solution to be applied via (B2). The exponentiation of a general matrix, even a tridiagonal one, is a non-trivial exercise, even with a software package such as *Mathematica* (which we have used, as did Graziani). The process is simplified tremendously by decomposing Λ into a sum of "even" and "odd" matrices which are almost block diagonal. As discussed by Graziani, the decomposition is not unique, and the effects of choosing one over another are not presently known. Nonetheless, we follow the method in Graziani's paper and define our decomposed matrices as

$$\Lambda_e = \begin{pmatrix} 0 & -(1-\theta_l)B & (1-\theta_l)B & 0 & 0 & 0 \\ 0 & -B & B & 0 & 0 & 0 \\ 0 & C & -C & 0 & 0 & 0 \\ 0 & 0 & 0 & -F & F & 0 \\ 0 & 0 & 0 & G & -G & 0 \\ 0 & 0 & 0 & (1-\theta_r)G & -(1-\theta_r)G & 0 \end{pmatrix}, \tag{B5}$$

and

$$\Lambda_o = \begin{pmatrix} \Gamma_l\theta_l & 0 & 0 & 0 & 0 & 0 \\ \theta_l A & -\theta_l A & 0 & 0 & 0 & 0 \\ 0 & 0 & -D & D & 0 & 0 \\ 0 & 0 & E & -E & 0 & 0 \\ 0 & 0 & 0 & 0 & -\theta_r H & \theta_r H \\ 0 & 0 & 0 & 0 & 0 & \Gamma_r\theta_r \end{pmatrix}. \tag{B6}$$

The exponentiation of Λ_e and Λ_o was performed using *Mathematica*, which proved to be highly useful in this exercise. We therefore arrive at our exponentiated matrices, which have the following form:

$$\exp(\Lambda_e) = \begin{pmatrix} 1 & -(1-\theta_l)\frac{B\left(1-e^{-(B+C)}\right)}{B+C} & (1-\theta_l)\frac{B\left(1-e^{-(B+C)}\right)}{B+C} & 0 & 0 & 0 \\ 0 & \frac{\left(C+Be^{-(B+C)}\right)}{B+C} & \frac{B\left(1-e^{-(B+C)}\right)}{B+C} & 0 & 0 & 0 \\ 0 & \frac{C\left(1-e^{-(B+C)}\right)}{B+C} & \frac{\left(B+Ce^{-(B+C)}\right)}{B+C} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\left(G+Fe^{-(F+G)}\right)}{F+G} & \frac{F\left(1-e^{-(F+G)}\right)}{F+G} & 0 \\ 0 & 0 & 0 & \frac{G\left(1-e^{-(F+G)}\right)}{F+G} & \frac{\left(G+Fe^{-(F+G)}\right)}{F+G} & 0 \\ 0 & 0 & 0 & (1-\theta_r)\frac{G\left(1-e^{-(F+G)}\right)}{F+G} & -(1-\theta_r)\frac{G\left(1-e^{-(F+G)}\right)}{F+G} & 1 \end{pmatrix}, \tag{B7}$$

and

$$\exp(\Lambda_o) = \begin{pmatrix} e^{\Gamma_l\theta_l} & 0 & 0 & 0 & 0 & 0 \\ \frac{A(e^{\theta_l(A+\Gamma_l)}-1)}{e^{\theta_l A}(A+\Gamma_l)} & e^{-\theta_l A} & 00 & 0 & 0 & \\ 0 & 0 & \frac{E+De^{-(D+E)}}{D+E} & \frac{D(1-e^{-(D+E)})}{D+E} & 0 & 0 \\ 0 & 0 & \frac{E(1-e^{-(D+E)})}{D+E} & \frac{D+Ee^{-(D+E)}}{D+E} & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{-\theta_r H} & \frac{H(e^{\theta_r(H+\Gamma_r)}-1)}{e^{\theta_r H}(H+\Gamma_r)} \\ 0 & 0 & 0 & 0 & 0 & e^{\theta_r\Gamma_r} \end{pmatrix}. \tag{B8}$$

In the case of symmetric matrices (which obtain from uniform cartesian grids), the original matrix elements become symmetric. In this instance, the matrix elements in (B7) and (B8) reduce to the corresponding expressions for the symmetric case presented in the Graziani paper.

With these matrices in hand, the PF solution is carried out by the technique outlined in the Graziani paper and which is identical to that in the original algorithm supplied to us by Graziani. What is needed to complete our discussion are the formulae for the matrix elements themselves, specifically the subdiagonal and superdiagonal elements employed for sweeps along each coordinate axis. These expressions, which include the timestep factor from the exponential term in (B2), may be listed as follows:

$$\text{subdiag(l)} = \begin{cases} \Delta t \cdot D1(i) \times \frac{g_2^a(i)g_{31}^a(i)}{\Delta V_1^a(i)\Delta x_1^a(i)} & \text{1-coordinate; l = i} \\ \Delta t \cdot D2(j) \times \frac{g_{32}^a(j)}{\Delta V_2^a(j)\Delta x_2^b(j)(g_2^b(i))^2} & \text{2-coordinate; l = j} \\ \Delta t \cdot D3(k) \times \frac{1}{\Delta V_3^a(k)\Delta x_3^b(k)(g_{31}^b(i)g_{32}^b(j))^2} & \text{3-coordinate; l = k} \end{cases} \tag{B9}$$

$$\text{superdiag(l)} = \begin{cases} \Delta t \cdot D1(i+1) \times \frac{g_2^a(i+1)g_{31}^a(i+1)}{\Delta V_1^a(i)\Delta x_1^a(i+1)} & \text{1-coordinate; l = i} \\ \Delta t \cdot D2(j+1) \times \frac{g_{32}^a(j+1)}{\Delta V_2^a(j)\Delta x_2^b(j+1)(g_2^b(i))^2} & \text{2-coordinate; l = j} \\ \Delta t \cdot D3(k+1) \times \frac{1}{\Delta V_3^a(k)\Delta x_3^b(k+1)(g_{31}^b(i)g_{32}^b(j))^2} & \text{3-coordinate; l = k} \end{cases} \tag{B10}$$

C. The Implicit Radiation Energy Equation Matrix

The matrix solved in the implicit diffusion algorithm is generated from a discretization of equation (16). Only the diffusion term contributes off-diagonal elements to the matrix, as the quantities multiplying E in the Eddington tensor term, $\mathcal{F}E$, are taken from the previous timestep. Thus most of the complexity arises from expanding $\nabla \cdot D\nabla E$, where D is a diffusion coefficient. Before proceeding, we note that in the code, (16) is symmetrized by multiplying both sides by a differential volume factor, which is written with the code variables defined in Appendix A as dvol1a(i)·dvol2a(j)·dvol3a(k). This expression is merely a product of the three volume differences between interfaces along the i,j, and k coordinate axes, respectively. For a given (i,j,k) mesh point, the main diagonal matrix term will be the sum of the identify matrix, the matter coupling term, the Eddington tensor term, and those terms of the diffusion operator involving E(i,j,k). All off-diagonal terms will consist strictly of the appropriate remaining pieces of the diffusion operator. Since the matrix is symmetrized, we need concern ourselves only with the upper off-diagonals in addition to the main diagonal. The first, second, and third superdiagonals will consist of those parts of the diffusion operator multiplying E(i+1,j,k), E(i,j+1,k), and E(i,j,k+1), respectively. Labeling the main diagonal of the matrix as DD(i,j,k), and refering back to (16), we may write

$$\begin{aligned} \mathrm{DD(i,j,k)} &= \mathrm{dvl1a(i) \cdot dvl2a(j) \cdot dvl3a(k)} \\ &\times \left(1.0 + \mathrm{cpl(i,j,k)} + dt \cdot \mathrm{grdvcf(i,j,k)} - \mathrm{oper(i,j,k)}\right), \end{aligned} \tag{C1}$$

where cpl(i,j,k), grdvcf(i,j,k), and oper(i,j,k) are the coupling, radiation stress, and diffusion operator terms, respectively. The radiation stress term is the tensor inner product of the velocity gradient (computed from the velocity at the previous timestep), and the Eddington tensor; i.e. $\nabla v : f$. The 3-D expression of this term is extremely lengthy and will not be included here, but will be supplied to interested readers upon request. An expression for the 2-D case may be found in Stone, Mihalas, and Norman (1992). The essential feature is that the expression is linear in E(i,j,k) alone. The cpl(i,j,k) term is evaluated as follows:

$$\mathrm{cpl(i,j,k)} = \frac{\mathrm{kre(i,j,k)}}{\Delta\mathrm{(i,j,k)}}, \tag{C2}$$

where

$$\mathrm{kre(i,j,k)} \equiv c \cdot dt \cdot \kappa_P; \tag{C3}$$

$$\Delta\mathrm{(i,j,k)} \equiv 1 + \frac{4\sigma \mathrm{kre(i,j,k)} T^3\mathrm{(i,j,k)}}{c_v \rho\mathrm{(i,j,k)}} \tag{C4}$$

The remaining contribution to DD(i,j,k) is that of oper(i,j,k), which is composed of the terms from $\nabla \cdot F$ which multiply $E(i,j,k)$, and is written as follows:

$$\begin{aligned}
\mathrm{oper(i,j,k)} \;=\; & dt \times \\
& [\\
& \quad - \mathrm{dvl1ai(i)} \cdot \\
& \quad [(\mathrm{g2a(i+1)} \cdot \mathrm{g31a(i+1)})^2 \cdot D1\mathrm{(i+1,j,k)} \cdot \mathrm{dvl1bi(i+1)} + \\
& \quad (\mathrm{g2a(i)} \cdot \mathrm{g31a(i)})^2 \cdot D1\mathrm{(i,j,k)} \cdot \mathrm{dvl1bi(i)}] \\
& \\
& \quad - \mathrm{dvl1ai(i)} \cdot \mathrm{g2bi(i)}^2 \cdot \\
& \quad [\mathrm{g32a(j+1)}^2 \cdot D2\mathrm{(i,j+1,k)} \cdot \mathrm{dvl2bi(j+1)} + \\
& \quad \mathrm{g32a(j)}^2 \cdot D2\mathrm{(i,j,k)} \cdot \mathrm{dvl2bi(j)}] \\
& \\
& \quad - \mathrm{dvl3ai(k)} \cdot \mathrm{g31bi(i)}^2 \cdot \mathrm{g32bi(j)}^2 \cdot \\
& \quad [D3\mathrm{(i,j,k+1)} \cdot \mathrm{dvl3bi(k+1} + D3\mathrm{(i,j,k)} \cdot \mathrm{dvl3bi(k)} \\
&]
\end{aligned} \tag{C5}$$

The superdiagonals, DDP1(i,j,k), DDP2(i,j,k), and DDP3(i,j,k), are more simply expressed and written as follows:

$$\begin{aligned}
\mathrm{DDP1(i,j,k)} \;=\; & -dt \cdot \mathrm{dvl2a(j)} \cdot \mathrm{dvl3a(k)} \cdot (\mathrm{g2a(i+1)} \cdot \mathrm{g31a(i+1)})^2 \cdot \\
& D1\mathrm{(i+1,j,k)} \cdot \mathrm{dvl1bi(i+1)}
\end{aligned} \tag{C6}$$

$$\begin{aligned}
\mathrm{DDP2(i,j,k)} \;=\; & -dt \cdot \mathrm{dvl1a(i)} \cdot \mathrm{dvl3a(k)} \cdot (\mathrm{g2bi(i+1)} \cdot \mathrm{g32a(i+1)})^2 \cdot \\
& D2\mathrm{(i,j+1,k)} \cdot \mathrm{dvl2bi(j+1)}
\end{aligned} \tag{C7}$$

$$\begin{aligned}
\mathrm{DDP3(i,j,k)} \;=\; & -dt \cdot \mathrm{dvl1a(i)} \cdot \mathrm{dvl2a(j)} \cdot (\mathrm{g31bi(i)} \cdot \mathrm{g32bi(j)})^2 \cdot \\
& D3\mathrm{(i,j,k+1)} \cdot \mathrm{dvl3bi(k+1)}
\end{aligned} \tag{C8}$$

In equations C5 through C8, $D1$, $D2$, and $D3$ refer to the face-centered diffusion coefficients along each of the three coordinate axes.

REFERENCES

Baldwin, C., Brown, P., Falgout, R., Jones, J., & Graziani, F. 1998, *J. Comp. Phys.*, submitted.

Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., & Van der Vorst, H. 1994, "Templates for the Solutoin of Linear Systems: Building Blocks for Iterative Methods," pub. SIAM.

Fiedler, R. 1997, http://www.supercomp.org/sc97/program/TECH/FIEDLER/INDEX.HTM

Graziani, F. 1995, *J. Comp. Phys.*, **118**, 9-23.

Stone, J. & Norman, M. 1992a, *Astrophys. J. Suppl.*, **80**, 753-790.

Stone, J. & Norman, M. 1992b, *Astrophys. J. Suppl.*, **80**, 791-818.

Stone, J., Mihalas, D., & Norman, M. 1992, *Astrophys. J. Suppl.*, **80**, 819-845.